Scientific Excellence • Resource Protection & Conservation • Benefits for Canadians
Excellence scientifique • Protection et conservation des ressources • Bénéfices aux Canadiens

A Model for Evaluation of Some Management Policy Options for Lobster or Moult-Type Fisheries

R. Shotton

Program Coordination and Economics Branch
Scotia—Fundy Region
Department of Fisheries and Oceans
P. O. Box 550
Halifax, Nova Scotia
B3J 2S7

July 1989

Canadian Technical Report of Fisheries and Aquatic Sciences 1696

Fisheries and Oceans Pêches et Océans

Canada

Canadian Technical Report of Fisheries and Aquatic Sciences

Technical reports contain scientific and technical information that contributes to existing knowledge but which is not normally appropriate for primary literature. Technical reports are directed primarily toward a worldwide audience and have an international distribution. No restriction is placed on subject matter and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries and aquatic sciences.

Technical reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report is abstracted in *Aquatic Sciences and Fisheries Abstracts* and indexed in the Department's annual index to scientific and technical publications.

Numbers 1–456 in this series were issued as Technical Reports of the Fisheries Research Board of Canada. Numbers 457–714 were issued as Department of the Environment, Fisheries and Marine Service, Research and Development Directorate Technical Reports. Numbers 715–924 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service, Technical Reports. The current series name was changed with report number 925.

Technical reports are produced regionally but are numbered nationally. Requests for individual reports will be filled by the issuing establishment listed on the front cover and title page. Out-of-stock reports will be supplied for a fee by commercial agents.

Rapport technique canadien des sciences halieutiques et aquatiques

Les rapports techniques contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles, mais qui ne sont pas normalement appropriés pour la publication dans un journal scientifique. Les rapports techniques sont destinés essentiellement à un public international et ils sont distribués à cet échelon. Il n'y a aucune restriction quant au sujet; de fait, la série reflète la vaste gamme des intérêts et des politiques du ministère des Pêches et des Océans, c'est-à-dire les sciences halieutiques et aquatiques.

Les rapports techniques peuvent être cités comme des publications complètes. Le titre exact paraît au-dessus du résumé de chaque rapport. Les rapports techniques sont résumés dans la revue *Résumés des sciences aquatiques et halieutiques*, et ils sont classés dans l'index annuel des publications scientifiques et techniques du Ministère.

Les numéros 1 à 456 de cette série ont été publiés à titre de rapports techniques de l'Office des recherches sur les pêcheries du Canada. Les numéros 457 à 714 sont parus à titre de rapports techniques de la Direction générale de la recherche et du développement, Service des pêches et de la mer, ministère de l'Environnement. Les numéros 715 à 924 ont été publiés à titre de rapports techniques du Service des pêches et de la mer, ministère des Pêches et de l'Environnement. Le nom actuel de la série a été établi lors de la parution du numéro 925.

Les rapports techniques sont produits à l'échelon régional, mais numérotés à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page du titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

Canadian Technical Report of

Fisheries and Aquatic Sciences 1696

July 1989

A MODEL FOR EVALUATION OF SOME MANAGEMENT POLICY OPTIONS FOR LOBSTER OR MOULT-TYPE FISHERIES

by

R. Shotton

Program Coordination and Economics Branch

Scotia-Fundy Region

Department of Fisheries and Oceans

P.O. Box 550

Halifax, Nova Scotia

B3J 2S7

89-05180/

(c) Minister of Supply and Services Canada 1989

Cat. No. 97-6/1696 E ISSN 0706-6457

Correct citation for this publication:

Shotton, R., 1989. A model for evaluation of management policy options for lobster or moult-type fisheries. Can. Tech. Rep. Fish. Aquat. Sci 1696: 52p.

ABSTRACT

A discrete-difference model for moult-type fisheries based on Scotia-Fundy Lobster fisheries is described. The objective of the model is to enable management options to be evaluated in terms of future landings, and future (discounted) revenues. Any number of management regulations can thus be compared through an estimate of their "net present revenues".

The model also enables the age structure of lobster populations to be determined after successive iterations together with the relative yields in weight as a function of lobster carapace length given different management scenarios. These results are used to determine annual yields and revenues during successive years.

The principal model control variables are cullsize, (i.e., carapace length at first possible capture or retention), and the annual rate of exploitation. The model is parameterized with sex-dependent functions of annual moult probability, weight and maturity; all functions of size. A population size-frequency structure is needed to initialize the model.

Keywords: Moult-fisheries, lobster, lobster fisheries, fisheries policy.

RÉSUMÉ

On décrit un modèle à différences discrètes pour les pêcheries d'espèces à mue fondé sur la pêche du homard dans la région de Scotia-Fundy. Le modèle en question vise à permettre l'évaluation des options de gestions en égard aux débarquements futurs et aux recettes futures (accualisées). On peut comparer de la sorte n'importe quel nombre de règlements de gestion au moyen d'une estimation des recettes actuelles obtenues sous leur régime.

Le modèle permet également de déterminer la structure d'âge des populations de homard après des itérations successives et les rendements relatifs en poids, fonctions de la longueur de la carapace, dans différents scénarios de gestion. Les résultats obtenus servent à déterminer les recettes et les rendements annuels successifs.

Les principales variables de contrôle du modèle sont la taille de sélection (p. ex.: longueur de la carapace à la première capture possible ou à la rétention) et le taux annuel d'exploitation. Le modèle est paramétré selon des fonctions qui dépendent du sexe et également de la taille: probabilité de mue annuelle, poids et maturité. On a besoin de connaître la fréquence par taille de la population pour initialiser le modèle.

Mots-clés: Espèces à mue, homard, pêche du homard, politique sur les pêches.

TABLE OF CONTENTS

	Abstract/Resumé	iii
1.	Introduction	1
2.	Population Dynamics	2
3.	Model Parameterization	5
	3.1 Introduction	5
	3.2 Mortality	7
	3.2.1 Natural Mortality	7
	3.2.2 Fishing Mortality	7
	3.3 Growth Functions	7
	3.4 Moult frequency	7
	3.5 Female Maturation	8
	3.6 Size-Weight Functions	8
	3.7 Initial Age-Size Population Structure	8
	3.8 Annual Recruitment	8
	3.9 Fishing Seasons	8
	3.10 Grade Prices	8
4.	Biological Model Structure	10
	4.1 Introduction	10
	4.2 GET_PARAMETERS	10
	4.3 SET_VARIALBES	10
	4.4 SET_POP_ARRAYS_TO_ZERO	12
	4.5 RECRUIT	12
	4.6 INITIALIZE_MALES and INITIALIZE_FEMALES	12
	4.7 OVIGEROUS	12
	4.8 SET_YIELDS_TO_ZERO	14
	4.9 HARVESTM	14
	4.10 HARVESTF	14
	4.11 HARVEST_TO_FILE	14
	4.12 WRITENUMBERS	14
	4.13 MOULT	14
5.	Revenue Model Structure	15
	5.1 Introduction	15
	5.2 GET_FILE	17
	5.3 SET_VARIABLES	17
	5.4 GET_DATA	17
	5.5 REVENUES	17
	5.6 WEIGHT	21
	5.7 "Rolling Wave" Moult Model	21
6.	Literature Cited	21

Appendices

I	Program Listing of the Biological Model	24
II	Revenues Program	33

IIIa Code for "Rolling Wave" Moult Model 37
IIIb Unit declaring Variables Used by LOBFLSH.PAS 48
IV Revenue Model (LOBFLSH.PAS) 51

LIST OF TABLES

1. Type Declared Constants 6
2. Program Declared Constants 6
3. Probability of Moult Models 9
4. Probability of Maturity: Females 9
5. An Examples of Output from the Biological Model 16
6. An Example of Output from the Revenue Model 19
7. An Example of Output from LOB2 Revenue Model 22
8. Output Summary from the Revenue Model LOB2 23

LIST OF FIGURES

1. Fishing Season Schematic 3
2. Main Program Procedure Structure 11
3. Program Logic for Maturing or Mature Females 13
4. Program Logic for the Revenue Module 18

1. Introduction

Formulation and evaluation of policy options for the management of a fishery requires that the respective costs and benefits of the different options can be determined. In this way, either the requirements for an optimal policy may be determined, and/or the costs of a sub-optimal policy identified. To do this, a model that can determine the yields and revenues for different values of the control variables is usually required. In lobster fisheries, these usually are the minimum carapace size at which a lobster can be retained, the total number of traps allowed (i.e., the product of number of licensed fishermen and the trap limit per licence), and the length of the fishing season.

A management action that increases future benefits from a fishery by reducing immediate and near-term catches will change the temporal distribution of cash flow. Thus additional, but deferred benefits should be discounted to account for the fact that future additional revenues will not accrue for an elapsed time period. What are appropriate social discount values is the subject of an extensive literature (e.g., Treasury Board 1982); in this model 10% is used but any value may be specified.

The model described here is a conventional discrete-difference yield-per-recruit model that accounts for growth by size specific moult increments. In this regard, the model is similar to that developed by Caddy (1979) and subsequently modified by Campbell (1985), but differs in that trajectories of yield can be followed as conditions (e.g. recruitment, cullsize, rate of exploitation, etc.) are changed over successive years. In this regard, the model outputs appear similar to those obtainable from the model described by Ennis and Akenhead (1978). A second program uses the yield by lobster size classes from successive annual harvests to determine revenues by size grades for the successive years, both in constant, and discounted, dollar values.

The models are written in Borland's Turbo Pascal, version 4, in contrast to the models of Caddy, Campbell, Ennis and Akenhead, which are written in Fortran. The model should run on any PC-AT that supports Turbo-Pascal. Because of the software's powerful text editting facilities, programming is easier if parameters are changed directly in the program using the edit option of the software's integrated environment facility. Despite this, most basic control variables are set by a user prompt and response procedure.

The number of possible model control combinations that one may wish to investigate is large, e.g., changes, or trends, in recruitment, growth, fishing mortality, rate of maturation, etc., during a simulation. To attempt to handle even some of these through a menu-prompt manner would make program use awkward. However, because of the structured design of the program, users with a basic knowledge of Turbo-Pascal should have little difficulty in modifying the software to obtain particular outputs they may require. Users wishing to do this are encouraged to contact the author for assistance.

In lobster fisheries, regulations often protect egg-bearing lobsters. This is done on the assumption that reduced exploitation of these lobsters will contribute to future recruitment. Thus this model handles males and females separately. Female lobsters, as modeled, may be immature, mature and non-berried or berried.

2. Population Dynamics

An arbitrary number of lobsters, separated by sex, $M_{1,s}$ (males of age one, and size s) and $I_{1,s}$ (immature females of age one, size s) are recruited to the fishery as a simulated population. These lobsters are allocated to a range in size, from the user-specified minimum fishable size to the next smallest size to that at which a minimum sized lobster becomes on moulting. For example, if the minimum length at capture is 81 mm, and such a lobster becomes 94 mm after moulting, then the initial population number is divided into 13 size classes for their first year in the simulation, i.e., 94 - 81 = 13 size classes in the first age group, or,

$$M_{1,size} = \frac{\text{initial number of recruits}}{L_{2,s} - L_{1,s}}$$

where $L_{1,s}$ = minimum size considered in model.
$L_{2,s}$ = size of crustacean after moulting, whose premoult size is $L_{1,s}$.

With a minimum (model) size at first capture of 81 mm and corresponding post moult size of 94 mm for a lobster with a premoult size of 81mm, and 100 000 recruits, then the numbers at age 1 in the size classes 81-93 would be,

$$M_{1,s} = \frac{100\ 000}{94 - 81} = 7692$$

During the fishing season an age class is subjected to a total mortality;

$$Z_f = (M_d + F_d)\ s$$

where: M_d = daily instantaneous natural mortality rate,
F_d = daily instantaneous fishing mortality rate,
s = length of fishing season in days.

The number of animals in a group that survive to the end of the fishing season will be,

$$N_f = N\ e^{Z_f}$$

where,

N = initial number of animals.

During the closed season, the group will be subject to only natural mortality, and the instantaneous natural mortality during this period will be,

$$M_c = M_d\ (365 - s)$$

The number of animals at the start of the second year will be,

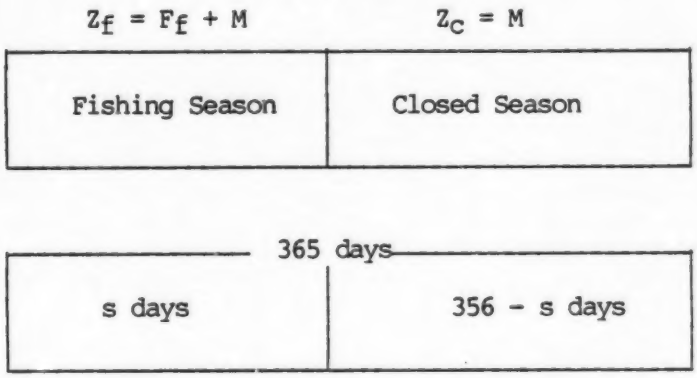
$$N_2 = N_f e^{-M_c}$$

Thus,

$$N_{i+1} = N_i\ e^{-Z_f} e^{-M_c}$$

The model starts at the opening of the fishing season with a full complement of recruits then a closed season follows. Moulting occurs at the end of the closed season. Figure 1 shows a schematic of this process.

Figure 1



For those lobster that in year y have an age class of "age" a, and size s, on transistion to their next year,

$$N_{y+1,s+1} = N_{y,s}\, P_s$$

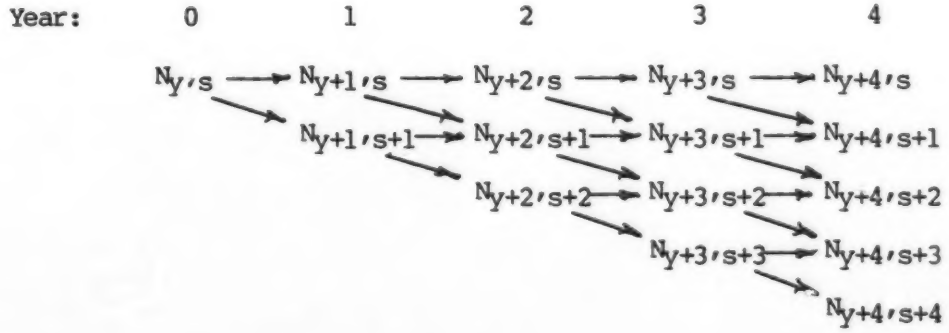
In this case a fraction P_s of the animals moult, increasing in size to s+1. For those that do not moult,

$$N_{y+1,s} = (1 - P_s)\, N_{y,s}$$

where,

P_s = fraction of animals of size s that moult,
$N_{y+1,s+1}$ = numbers oflobsters of size s that moult to the next size s+1, in year y + 1,
$N_{y+1,s}$ = numbers of animals that do not moult on becoming a year older and thus remain the same size.

Thus the numbers of animals that recruit in year zero (indicated as subscript y) change in size as follows:



The size of an animal at the n^{th} moult is that size attained after that number of moults by an animal with an initial size s.
The number caught at any age and size is:

$$_cN_{y,s} = (1 - e^{Zf})\ N_{y,s}$$

An unequivocal interpretation of models descibing the rate at which lobsters moult is difficult. For example, Campbell (1985) gives models that determine the "proportions moulting annually at different carapace lengths." But what subsequently happens to the lobsters that do not moult at a given size is not clear. If the proportion of N_y lobsters moulting one year is P, does that imply that of the animals that do not moult, N_y (1 - P) P moult the following year, or alternatively, is it N_y P (or N_y [1 - P] if P is greater than 0.5)? Further, it is most likely that the proportion of lobsters of a given age and size mouting in a year changes with age; insufficient biological information is available to explicitly answer these questions. Code for both moult models is described in Section 4.13.

The yield in kg is obtained from the numbers caught using a weight-at-size function (see Section 3.6),

$$Y_{y,s} = {}_cN_{y,s}\ W_s$$

where $$W = a\ s^b$$

The yield at size is required for the revenue model because the price/lb received for lobster depends of their size.

The proportion of females that are mature at a given size increases with size. After each moult, the expected number of mature animals at a size s, is determined by

$$N_{m,s} = p_o\ (I_s + O_s + B_s)$$

where

$N_{m,s}$ = expected number of mature animals at size s,
p_o = probability of a female being mature at size s,
I_s = number of immature animals at size s,
O_s = number of mature non berried animals at size s,
B_s = number of berried animals at size s.

Thus,

$$I_s = N_{m,s} - O_s - B_s$$

In this model, mature lobsters in year y, become berried in year y +1, i.e.,

$$B_{y+1} = O_y$$

The number of mature, but non-berried lobsters at size s is thus

$$O_{y+1,s} = N_{m,y,s} - B_{y+1,s}$$

Implicit in this is that lobsters first carry eggs in their second year of maturity, and are then berried in alternate years. These assumptions are not held in all situations but are considered reasonable for Scotia-Fundy.

3. Model Parameterization

3.1 Introduction

The program is parameterized in two ways; (1) type declared constants (a Pascal feature, see Table 1 and, (2) user entered constants. Deciding what should be a menu-prompted parameter and what a declared constant is a little arbitrary as it depends on the focus of the research, e.g., one analysis may be concerned with the cullsize chagnes keeping fishing mortality constant; in another study, cullsize may remain constant while another parameter is varied.

Requiring all parameters to be user-entered would make the program unwieldly. However, the reverse is not true as the "Integrated Environment" of Turbo Pascal makes program changes easy. Thus large amounts of code necessary to prompt for, and check input, can be avoided. A compromise, with parameters being declared both ways has been used in the program. Non-trivial users should aim to operate entirely within the integrated environment mode for parameter specification.

The user-entered variables in the biological model are:

Julian date of opening of fishing season,
Julian date of season closure,
Annual rate of exploitation in period prior to simulation,
Annual rate of exploitation during simulation,
Cullsize prior to simulation period,
Cullsize during simulated period.

All other parameters are declared as fixed or variable constants.

The third manner of program initiatization is through program code. For example, Campbell (1985) gives as a model for growth per moult,

$$L_{y+1} = yL^b \qquad (1)$$

In an earlier paper (1983) he uses a linear model,

$$L_{y+1} = y + bL \qquad (2)$$

In Pascal, Equation 1 is coded (for males) as

```
NextM[size] = round (exp(ln(amoultM) + bmoultM * ln (size))).
```

Equation 2 is coded (for females) as

```
NextF[size] = round(amoultF[size] + bmoultF[size] * size)
```

To switch from one to another, both code could be included and brackets[1] used to "comment out", that which is desired to be null code.

1/ Braces are used to create comments in Pascal, such code is not compiled.

Table 1

Type Declare Constants

Lines 1 - 13 show examples of type declared constants in the main program.

```
PROGRAM LCB35;     [LFAS 35, 36, 38, two cullsizes, 19 Dec 1988]
[$R+,$U+]
Uses Dos,Printer,Crt,Listnums;
1     Const
2          amoultM : real = 1.717;
3          bmoultM : real = 0.912;
4          amoultF : real = 2.107;
5          bmoultF : real = 0.864;
6          apropM : real = 1.76;
7          bpropM : real = -0.0084;
8          apropF : real = 1.95;
9          bpropF : real = -0.0097;
10         awtM : real = 0.0000566;
11         bwtM : real = 3.078;
12         awtF : real = 0.001525;
13         bwtF : real = 2.8612;
14         M_annual : real = 0.1;
```

Table 2

Examples of Program Declared Constants

These can be subsequently changed in the main program. For example, if the fishery subsequently took canners, the start size could be changed to 70mm.

```
15         Firstyear = 1989;
16         Lastyear = 1998;
17         Startsize = 81;
18         maxage = 5;
19         maxsize = 160;
```

3.2 Mortality

3.2.1 Natural Mortality

As is common for most fisheries, the natural mortality (M) of American lobster (Homarus americanus) is not well determined. The most commonly quoted value is 0.1 (e.g., Caddy 1979, Campbell 1985). The basis for this value is the analysis of Thomas (1973). However Botsford et al. (1986: 72-73) note that there are substantial reasons to doubt the methods used to obtain them, and that "their questionable nature is an important issue in light of the critical dependence of yield-per-recruit results on them."

Botsford et al., in their analysis, use a value of M = 0.15, but with no other justification. Vetter (1988) reviews the consequences for fishery models of incorrect choice of M. In general, higher estimates of M lead to lower estimates of maximum yield or maximum possible yield-per-recruit, higher estimates of the optimal fishing mortality (i.e. catch them before they die), and lower estimates of the optimal size at first capture. Botsford et al. (1988) found that using an M = 0.18 gave a long term [annual] increase in yield with their model of 6.8%, but this fell to 4.8% if an M = 0.2 was used.

3.2.2 Fishing Mortality

Values for fishing mortality are required to initialize the age-size structure population, and as such would represent the rate of fishing applied to the population prior to the start of the simulation. Fishing mortality is specified in the model as the annual rate of exploitation (A). The annual instantaneous fishing mortality is then obtained by,

$$F = - \ln (1 - A)$$

3.3 Growth Functions

Two forms of moult functions are commonly specified for the American lobster. An exponential model is given by Campbell (1985) for the Bay of Fundy:

Males: $L_{m+1} = 1.717 L_m^{0.912}$, and

Females $L_{m+1} = 2.107 L_m^{0.864}$.

Linear models specified by Campbell (1983) for the Bay of Fundy are:

Males: $L_{m+1} = 10.10 + 1.04 L_m$

Females: immature $L_{m+1} = 9.60 + 1.04 Lm$

Females, mature $L_{m+1} = 18.0 + 0.95 Lm$

Miller, Moore and Pringle (1987, Table 1) list other coefficients for linear lobster growth models.

3.4 Moult Frequency

Table 3 lists moult frequency models, by Lobster Fishing Area. Note that these are interpretted as a proportional model.

3.5 Female Maturation

Table 4 lists models and coefficients for the probability of maturity for female lobsters as a function of size.

3.6 Size-Weight Functions

Size-weight functions for males and females are given by Campbell (1985):

$$\text{Males} \quad W = 5.66 \times 10^{-7} \, L^{3.0780} \text{ (kg)}$$

$$\text{Females} \quad W = 1.525 \times 10^{-6} \, L^{2.8612} \text{ (kg)},$$

where lengths are measured in cm. Other length-weight functions are given by Miller et al. (1987).

3.7 Initial Population Age-Size Structure

A major objective of the model is to determine what happens to catches and revenues when a population that has been fished under conditions of one set of mortality and cull sizes is subsequently fished at a different rate of exploitation and minimum carapace size. To investigate this an initial frequency-at-size matrix is established using the appropriate cull size (which together with natural mortality determine size at entry to the fishery) and fishing mortality which determines numbers at subsequent ages. The size frequency matrix is then fished under the new conditions and the changes in yield characteristics determined.

Although the model has not been so written, it can be easily modified so that the initial population structure is taken from a real population, or any arbitrary size frequency distribution could be used. In this case, results for the first phase of the simulated exploitation could reflect how well (or poorly) the model assumptions describe the past fishery.

3.8 Recruitment

The model assumes a constant recruitment of 10 000 males and 10 000 females each year. The code can be simply changed to model any assumption about recruitment one wishes. For example, a stochastic process, or trend, or some combination, could be applied to the numbers that recruit each year to examine the consequences of different assumptions of variable recruitment. Multiple recruitment within a year would be possible but would require programming changes.

3.9 Fishing Seasons

The model is structured with a two phase annual cycle; first a fishing season, then a closed season. Moulting and increase in age occur at the end of the closed season. The model could be easily modified by an appropriate loop to handle any number of fishing and closed seasons within a year. An appropriate array would be necessary to store the open and close days, or equivalently the season lengths. These would then be accessed through an appropriate variable in the main program loop.

3.10 Grade Prices

Clearly any set of prices used for the differing sized lobsters will be arbitrary, especially as they will vary in response to changes in the relative supply of the different grades. Development of an acceptable demand-supply function should be the next objective of any future model extension.

TABLE 3

PROBABILITY OF MOULT MODELS

LFA		MODEL	SOURCE
33, 34-38	Males	$1.76e^{-0.0084L}$	Campbell (1985) or Miller et al. (1987).
	Females	$1.95e^{-0.0097L}$	Campbell (1985) or Miller et al. (1987).
28-32	Both sexes	$2.59e^{-0.014L}$	Miller et al 1987
27	Both sexes	$3.77e^{0.019L}$	Dube 1986

TABLE 4

PROBABILITY OF MATURITY; FEMALES

LFA	MODEL	SOURCE
34-38	$\frac{1}{1 + e^{23.23-0.214L}}$	Campbell and Robinson (1983) or Miller et al. (1987).
33	$\frac{1}{1 + e^{14.23-0.14L}}$	Campbell and Robinson (1983) cited by Miller et al. (1987).
28-31	$\frac{1}{1 + e^{10.4 - 0.112L}}$	Campbell and Robinson (1983)
27	$\frac{1}{1 + e^{21.8 - 0.258L}}$	Miller et al. (1987)

4. Biological Model Structure

4.1 Introduction

Figure 2 shows the relation of the different program modules. Procedures 1-6 are executed once to initialize the population; procedures 7-13 are executed for each year of the simulation. Full details of the program code are given in Appendix I.

4.2 GET PARAMETERS

This procedure handles the user-specified variable setting for the program. By successive prompts, the user is asked to enter:

1. A name for the simulation (up to 80 characters). This is echoed by the Fiscal module that processes the output from the biological model.

2. Julian dates of the opening and closing days of the fishing seasons. These should be entered as a pair separated by a blank.

3. The annual rate of exploitation to be used in initializing the population structure. This must have the form 0.nd where nd is any number of digits. This would be the same when initializing the population as when it was being exploited.

4. The annual rate of exploitation to be used in the simulation is also 0.nd

5. The cull size that is to be used in initializing the population structure. This would be the cull size used prior to any change in fishing regulations.

6. The future cull size.

7. The name of the output file. This must conform to DOS requirements, i.e., the name can have up to 8 characters plus a 3 letter extension separated by a period.

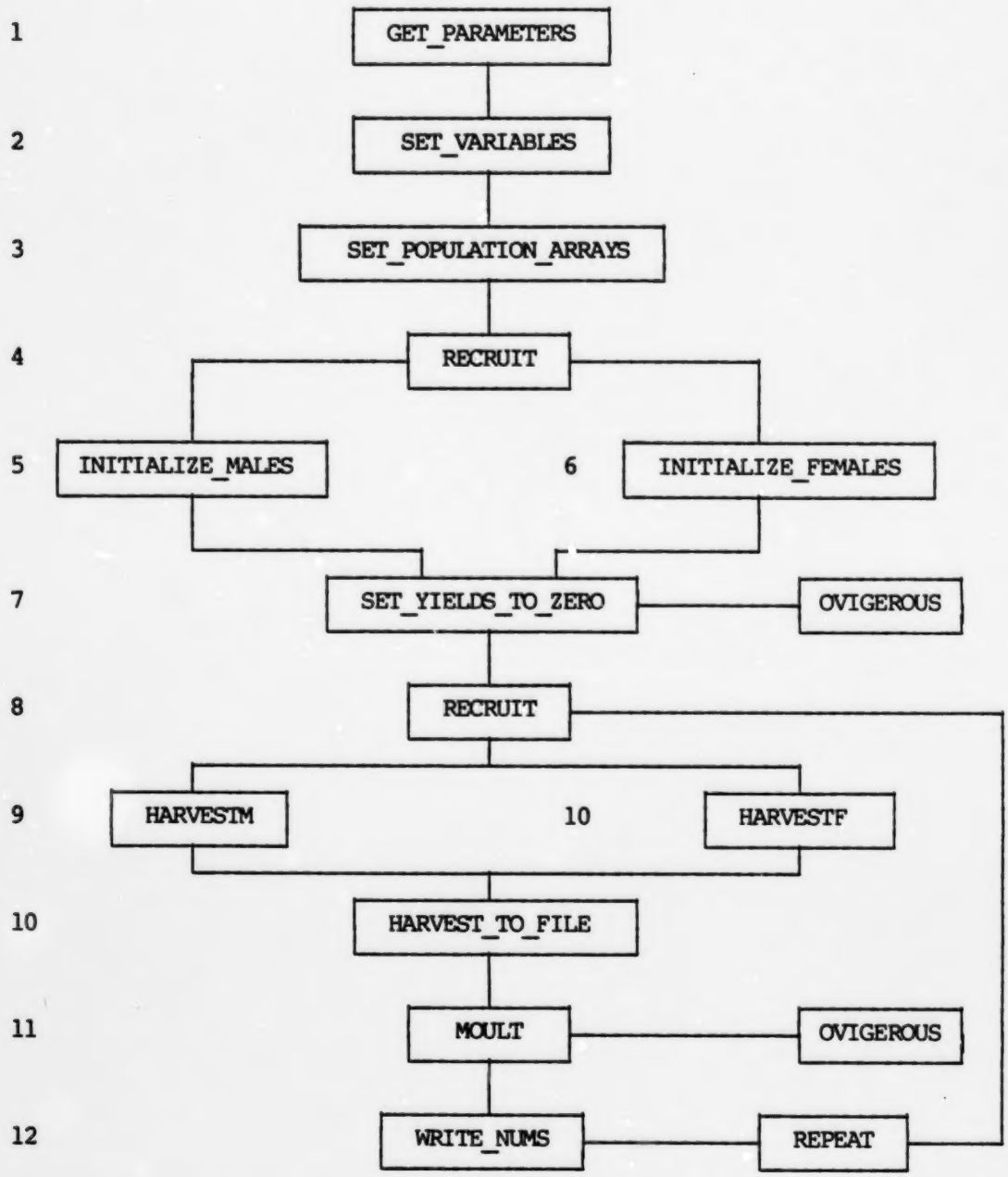
8. Length-age frequency distributions can be printed by appropriate responses to the prompt; the initial, final, or both, population structures can be listed.

4.3 SET VARIABLES

This procedure calculates the size of a lobster after its next moult. It also calculates the proportion of lobsters that will moult as a function of size. Unlike the output of the model of Ennis and Aikenhead (1977), there is a one-to-one correspondence between premoult and post-moult sizes. The model of Ennis and Aikenhead results in a prespecified distribution of post moult sizes for a given pre-moult size. The program could be easily modified to account for such a moult model if it could be parameterized.

Figure 2

Main Program Procedure Structure



4.4 SET-POP-ARRAYS-TO-ZERO

Turbo-Pascal does not initialize declared variables to zero. This procedure sets the age-size frequency arrays for male, immature female, mature female (non-berried) and berried females to zero.

4.5 RECRUIT

This procedure takes an arbitrary, but program specified, number of animals (10 000) and assigns them to the age one male and immature female age-size frequency arrays. The minimum size is specified by the constant "startsize" specified as a constant in the variable declaration code of the first procedure declaration. The next moult size for a lobster of startsize is determined (Nextsize), and the 10 000 lobsters are equally assigned among the (Nextsize - startsize) size frequencies. For example, with a startsize of 81mm, Nextsize [81] = 94 mm, thus 10 000 animals are allocated to the 81 to 93 mm size classes.

For females, the proportion of mature female lobsters in the size range startsize to NextsizeF-1 is next calculated, and the numbers of immature and mature females so determined. It is assumed that no berried lobsters of age one (i.e., the first year of the simulation) exist. The procedure RECRUIT is called in two places; first in the program initialization and then for each year of the simulation. Should any change in recruitment, e.g., a trend or stochastic process be desired, it could be handled in this procedure. Alternatively, frequencies otherwise determined could be read from a file.

4.6 INITIALIZE MALES and INITILAIZE FEMALES

These procedures establish a population structure using the variables Old_cull_size and Pre_rate_of_exploitation, that represents the conditions of the population prior to exploitation by the model. By determining the annual survivors, the numbers at successive ages are determined, and from the proportion moulting, the numbers in subsequent size classes. The number of age classes considered in the simulation is specified by the constant maxage, and the total number of size classes (in 1mm increments) by startsize - maxsize + 1. While survivors may occur in the largest age and size classes, under most conditions their numbers are negligible.

The INITIALIZE_FEMALES procedure differs from that for the males in that it determines the numbers of mature and berried females. The logic used to do this is shown in Figure 3. Mature females that survive alternate annually between being berried and non berried. Newly mature lobsters do not become berried in their first year of maturity. In reality, berried lobster may be berried in succesive years and immature lobster may be berried in the first year they mature (Per comm. D. Pezzack, Benthic Fisheries and Aquatic Division, Scotia-Fundy Region, Department of Fisheries and Oceans). However, the model as coded does not account for this. If parameters were available the program can be easily modified to account for such maturity transfomations.

During fishing seasons, lobsters are subject to natural and fishing mortality (F_f+M_f). Between fishing seasons, they are subject only to natural mortality (M_c).

4.7 OVIGEROUS

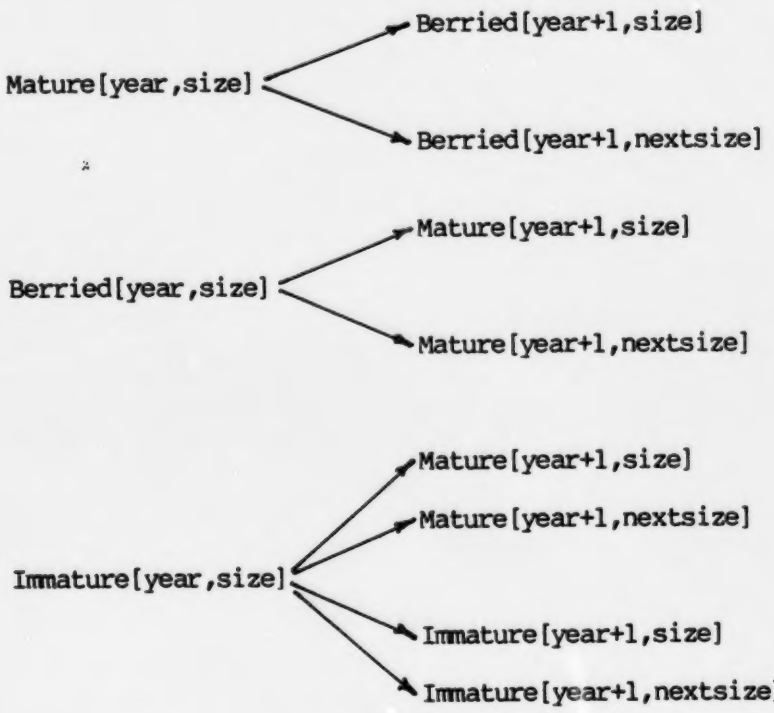
This procedure determines the numbers of lobsters that should be mature at a given size, and as necessary, reduces the number of immature females lobsters

Figure 3

Program Logic for Maturing or Mature Females

In RECRUIT:

In subsequent procedures:



and increases the number of mature female lobsters. It is called during model initialization and during each year of the model simulation. Because males are fished irrespective whether mature or immature, state of male lobster maturity is immaterial to the model.

4.8 SET YIELDS TO ZERO.

Prior to harvest, the arrays storing the yield in weight by size category and sex (male, immature female and mature female) are set to zero.

4.9 HARVESTM

This procedure harvests the male lobsters whose numbers by age and size are stored in the array Males. The conventional models are used:

$$\text{survivors} = \text{initial number} * e^{-Zf}.$$

Catch in numbers at the end of the season is determined by,

$$C = F_f / Z_f \ (1 - e^{-Zf}) * \text{initial numbers}$$

Should the lobster size be less than the cull size, the fishing mortality is set to zero; likewise it is set to zero for periods outside fishing seasons.

4.10 HARVESTF

This procedure is similar in function to HarvestM. The immature and mature lobsters are harvested and their numbers correspondingly reduced. The numbers of berried lobsters are subjected solely to natural mortality during the harvest period (fishing season). The numbers of lobsters harvested are multiplied by their weights at size and the catches of immature and mature lobsters are stored in the arrays YieldI[size] and YieldO[size] respectively.

4.11 HARVEST-TO-FILE

The procedure writes to the user-specified disc file the yield by weight for each size class of males, immature females and mature females.

4.12 WRITENUMBERS

This procedure, when called, writes the age-size frequencies to the line printer for each of the four lobster classes modeled; males, immature, mature and berried females. The procedure calls LIST-AGESIZE. Should it be desired that these numbers be written to disc, the writeln statements in LIST-AGESIZE should be modified.

4.13 MOULT

This procedure moves lobsters from one age and size to either the next age (a+1) and the same size (i.e., those lobsters that did not moult), or the next age and next size (s+1), i.e., for those lobsters that do moult, as determined by the moult model

A simple fractional moult model can be used. In this case, a specified fraction of all lobsters at the same size moult. Implicit in this model is that some lobsters of a particular year class never moult and their numbers in a year and size class simply decline from year to year from natural mortality. Thus if 60% of an age class moulted each year, the numbers at the initial size would be (ignoring mortality) as follows:

	Numbers at initital size	
Year 1	10 000	recruits
Year 2	4 000	survivors at age 2 and still at the initial size.
Year 3	1 600	survivors at age 3
Year 4	640	survivors at age 4
Year 5	256	survivors at age 5.

At the end of n years, $(1-p)^n$ lobsters will still have the same size (ignoring mortality) as in the year they recruited to the fishery.

An alternative moult model has been progammed in which the numbers of animals moulting each year is not a proportion of the remaining numbers at size but rather the proportion of the original numbers, adjusted for mortality. In the case above, with "flush" or "rolling wave" type moulting, a probability of moulting = 0.6 and ignoring mortality, then the numbers at age and size would be:

Year	Numbers at size
1	10 000
2	4 000
3	0

Should 30% of animals moult each year, then the numbers of a year class remaining at a particular size with a "rolling wave" type moult model (ignoring mortality) would be as follows:

Year	Numbers at size
1	10 000
2	7 000
3	4 000
4	1 000
5	0

In this model after [truncate(1 + 1/proportion moulting)] years, all lobsters would be flushed through to the next larger size class. This moult model involves considerably more coding as a shadow array must be maintained to track the numbers of lobsters that must moult for each year class entering the fishery. These shadow arrays, one for each class of lobster, must also be adjusted for natural and fishing mortality as appropriate.

In the case of female lobsters, the moult procedure is also where mature non-berried female lobsters are switched to berried lobsters at the end of the closed season, and berried lobsters are switched to mature non-berried female lobsters.

Table 5 shows an example of the output of the Biological Model. The code for this model is not given in this report.

5. Revenue Model Structure

5.1 Introduction

By separating the algorithms for calculating revenues from those required to determine yields by size, different market assumptions can be examined

Table 5

An example of the output from the Biological Model.

The first line is a comment line echoing run identification characters entered by the user. The data on the second line are: initial cull size, final cull size, maximum size in the simulation, year simulations start, and year simulation stops. The third line gives the year of the first annual results The subsequent rows give the numbers caught at the respective carapace sizes.

```
LFA 27, 2 rates of exploitation 70/85 days = 61
  70  70 160  1989  1998   -0.00
1989
 CL      Males     Immature    Mature
 70      166.7      174.6        4.1
 71      175.2      181.8        5.5
 72      184.1      188.8        7.4
 73      193.3      195.4        9.9
 74      202.8      201.5       13.1
 75      212.7      206.9       17.3
 76      222.9      211.3       22.8
 77      233.4      214.3       29.8
 78      244.3      215.7       38.6
 79      255.6      215.0       49.6
 80       91.7       72.8       19.9
 81      120.8       89.6       32.1
 82        0.0       84.9       38.8
 83      129.3       79.3       46.4
 84      132.8       72.9       54.4
 85      136.3       65.9       62.7
 86      139.8       58.7       70.8
 87      143.3       51.6       78.5
 88      146.9       44.7       85.5
 89      150.4       38.4       91.6
 90        0.0       11.5       35.3
 91      159.7        4.7       18.7
 92       56.0        4.0       19.9
```

without requiring the biological computations to be repeated. This model reads the file created by the biological model. Figure 3 shows the logic structure of the model. The procedures 2-5 are repeated twice; the first iteration analizes the status quo or initial variable conditions, the second iteration analizes the alternative set of management variables. An example of the output of this program is given in Table 6.

5.2 GET FILE

Procedure GET-FILE prompts for the name of the user-specified output file from the biological module. It assigns the file unit, then appends and resets it.

5.3 SET-VARIABLES

This procedure sets the arrays which accumulate the weights and revenues by grade to zero.

5.4 GET-DATA

This procedure reads the output from the biological model for processing. The first record reads the following variables:

minsize: minimum size at which the crustaceans enter the fishery.

cullsize: size of animals at which they are recruited to the fishery (Knife-edge recruitment).

maxsize: maximum age of animals considered by model. Animals older than maxage "fall off the edge" of the population matrix.

yearstart: the first year of the simulation.

yearstop: last year of the simulation.

Rate-of-Exploitation: Annual rate of exploitation used in biological model. Recorded for information purposes only.

Each subsequent line (from minsize to maxsize) consists of:
- cephalothorax size
- catch of males (kg) at this size
- catch of immature females (kg) at this size
- catch of mature females at this size

These lines are processed through a number of loops that correspond to the different grades:

canners	to 80 mm cephalothorax size,
chix	81 - 89 mm,
quarters	90 - 95 mm,
halves	94 - 104 mm,
selects	105 - 119 mm,
jumbos	greater than 120 mm.

5.5 REVENUES

This procedure assigns revenues to the respective grades. The prices per lb are declared as constants at the start of the "Revenues" procedure. Revenues

Figure 4

Program Logic for Revenue Model

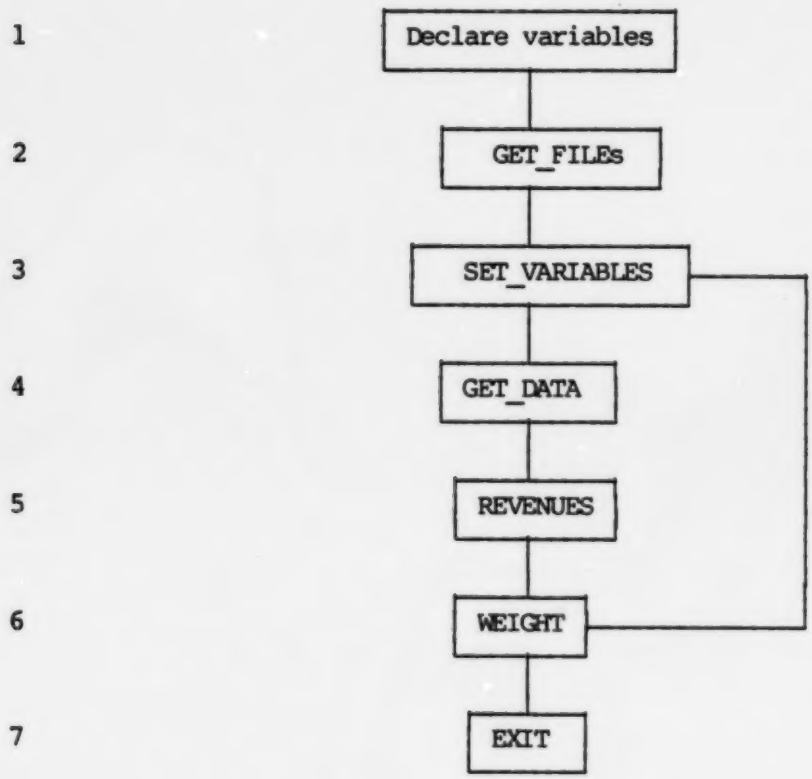


Table 6

An example of output from the Revenue Program.

LFA 34 days = 187 a = 0.83, 81/85

cullsize= 81

REVENUES

Year	Canners	Chix	Quarters	Halves	Selects	Jumbos	Total	Discounted
1	0	21625	13803	7124	4742	726	48020	48020
2	0	21625	13803	7124	4763	696	48012	43647
3	0	21625	13803	7126	4773	683	48010	39678
4	0	21625	13803	7126	4773	685	48012	36072
5	0	21625	13803	7126	4773	682	48010	32791
6	0	21625	13803	7126	4773	682	48010	29810
7	0	21625	13803	7126	4773	682	48010	27100
8	0	21625	13803	7126	4773	682	48010	24637
9	0	21625	13803	7126	4773	682	48010	22397
10	0	21625	13803	7126	4773	682	48010	20361
							480112	324513

Total discounted over a twenty year period 449621

WEIGHT

Year	Canners	Chix	Quarters	Halves	Selects	Jumbos	Total
1	0	5767	3451	1187	677	112	11194
2	0	5767	3451	1187	680	107	11192
3	0	5767	3451	1188	682	105	11192
4	0	5767	3451	1188	682	105	11192
5	0	5767	3451	1188	682	105	11192
6	0	5767	3451	1188	682	105	11192
7	0	5767	3451	1188	682	105	11192
8	0	5767	3451	1188	682	105	11192
9	0	5767	3451	1188	682	105	11192
10	0	5767	3451	1188	682	105	11192
							111923

cullsize= 85

REVENUES

Year	Canners	Chix	Quarters	Halves	Selects	Jumbos	Total	Discounted
1	0	12876	13803	7126	4773	682	39260	39260
2	0	12876	18434	14397	4773	682	51163	46512
3	0	12876	19060	15505	7682	682	55805	46120
4	0	12876	19130	15644	8239	1065	56955	42791
5	0	12876	19138	15661	8323	1300	57298	39135
6	0	12876	19138	15661	8323	1300	57298	35577
7	0	12876	19138	15661	8323	1300	57298	32343
8	0	12876	19138	15661	8323	1300	57298	29403
9	0	12876	19138	15661	8323	1300	57298	26730
10	0	12876	19138	15661	8323	1300	57298	24300
							546971	362172

Total discounted over a twenty year period 511484

Table 6 Continued

Year	Canners	Chix	WEIGHT Quarters	Halves	Selects	Jumbos	Total
1	0	3434	3451	1188	682	105	8859
2	0	3434	4609	2399	682	105	11229
3	0	3434	4765	2584	1057	105	11985
4	0	3434	4783	2607	1177	164	12164
5	0	3434	4784	2610	1189	200	12217
6	0	3434	4784	2610	1189	200	12217
7	0	3434	4784	2610	1189	200	12217
8	0	3434	4784	2610	1189	200	12217
9	0	3434	4784	2610	1189	200	12217
10	0	3434	4784	2610	1189	200	12217
							117540

are calculated by multiplying the catch in weight by grade by the corresponding prices, for each year. The annual revenues are calculated by summing these values. This procedure also discounts future revenues, i.e.,

$$R_i = R_1 (1 + d)^{-i},$$

where R_1 = revenue in first year of simulation,
R_i = revenue in the ith year,
d = user-specified discount rate.

The discount rate is declared as a constant in the main program. Two total discounted revenues are calculated, one for 10 years, the other for 20 years. The 10 year value is given by

$$T_{10} = \sum_{i=1}^{10} R_i (1 + d)^{-i}$$

The 20 year value is given by

$$T_{20} = T_{10} + \sum_{i=1}^{20} R_{10} (1 + d)^{-i}$$

Revenues are not calculated past 10 years. However, no matter what cull size increase is chosen, model outputs usually stabilize by year 4 or 5 when only one change in the cull size is undterken. Thus R_{10} can be safely taken as an equilibrium revenue value under the new management conditions. Table 6 shows an example of the revenue outputs from the model.

5.6 WEIGHT

This procedure lists the catch in kg by grade for the period of the simulation, nominally 10 years. Note that this represents the yield from 20 000 recruits, 10 000 of each sex.

5.7 Revenues from Rolling-Wave Moult Model

Table 7 lists output from the Program LOB2 (Appendix IV) which process the output from the "Rolling Wave" moult-type model (Appendix III). These results are slightly different to those from the Proportional Model in that the biological model calculates the catch and its composition for an increase in cull-size over a range of new cullsizes, e.g., from 81mm to 101mm. The program also lists the Net Present Value summed over 20 years for the succesive cull size options. This is shown in Table 8.

6. Literature Cited

Treasury Board 1982. Benefit-Cost Analysis Guide. Planning Branch, Treasury Board Secretariat. Ministry of Supply and Services.

Botsford, L.W., J.E.Wilen and E.J. Richardson, 1986. Biological and economic analysis of lobster fishery policies in Maine. L. W. Botsford and Associates. 98pp.

Caddy, J.F. 1979. Notes on a more generalized yield per recruit analysis for crustaceans using size-frequency inputs. Fish. Mar. Serv. Ms, Rep. No. 1525, 7pp.

Campbell, A.J. 1983. Growth of Tagged American Lobster, Homarus americanus, in the Bay of Fundy. Can. J. Fish. Aquat Sci. 40:1667-1675.

Campbell, A.J. 1985. Application of a Yield and Egg-per-Recruit model to the Lobster Fishery in the Bay of Fundy. N. Am. J. Fsih. Mgmt. 5:91-104.

Ennis, G.P. and S.A. Akenhead, 1978. A model and computer program used to assess yield per recruit in Newfoundland lobster stocks. CAFSAC Res. Doc. 78/30. 13pp.

Miller, R.J., D.S. Moore and J.D. Pringle 1987. Overview of the Inshore Lobster Resources in the Scotia-Fundy Region. CAFSAC Res. Doc. 87/85 20pp.

Thomas, J.C., 1973. The analysis of the commercial lobster (Homarus americanus) fishery along the coast of Maine, August 1968 through December 1970. Spec Sci. Rep. Fish. 667.

Vetter, E.F., 1988. Estimation of natural mortality in fish stocks: A Review. Fish Bull. 86(1):25-43.

Table 7

An example of output from LOB2, the revenue model for the "Rolling Wave" moult model. This model determines the fishery results for a sequence of cull sizes, e.g., from 81mm to 101mm.

SUMMARY OF 20 YEAR DISCOUNTED REVENUES BY CULLSIZE

cullsize (mm)	20 year discounted value
81	239680
82	242157
83	247175
84	254630
85	262201
86	269879
87	277645
88	285562
89	293580
90	301691
91	309010
92	317970
93	328496
94	339149
95	345784
96	348553
97	348931
98	349121
99	349109
100	348811
101	348285

Table 8

An example of output from LOB2, the revenue model for the "Rolling Wave" moult model. This part of the output summarizes the annual results disconted over a 20 year period after a change in the moult size.

REVENUES BY CATAGORIES for CULLSIZE = 81

Year	Chix	Quarters	Halves	Selects	Jumbos	Total	Discounted
1990	20908	13246	8664	6427	779	50024	50024
1991	20908	13246	8664	6427	786	50031	45482
1992	20908	13246	8664	6427	786	50031	41348
1993	20908	13246	8664	6427	786	50031	37589
1994	20908	13246	8664	6427	786	50031	34172
1995	20908	13246	8664	6427	786	50031	31065
							239680

20 year discounted total =

CATCH BY SIZE CATAGORY

Year	Chix	Quarters	Halves	Selects	Jumbos	Total
1990	5575	3311	1444	918	120	50024
1991	5575	3311	1444	918	121	50031
1992	5575	3311	1444	918	121	50031
1993	5575	3311	1444	918	121	50031
1994	5575	3311	1444	918	121	50031
1995	5575	3311	1444	918	121	50031

REVENUES BY CATAGORIES for CULLSIZE = 82

Year	Chix	Quarters	Halves	Selects	Jumbos	Total	Discounted
1990	18912	13246	8664	6427	779	48028	48028
1991	18912	15382	8664	6427	786	50171	45610
1992	18912	15696	8664	7215	786	51273	42374
1993	18912	15704	8664	7511	786	51577	38751
1994	18912	15704	8664	7576	816	51672	35293
1995	18912	15704	8664	7583	837	51700	32102
							242157

20 year discounted total =

CATCH BY SIZE CATAGORY

Year	Chix	Quarters	Halves	Selects	Jumbos	Total
1990	5043	3311	1444	918	120	48028
1991	5043	3846	1444	918	121	50171
1992	5043	3924	1444	1031	121	51273
1993	5043	3926	1444	1073	121	51577
1994	5043	3926	1444	1082	125	51672
1995	5043	3926	1444	1083	129	51700

Appendix 1

Program listing of the Biological Model.

```
PROGRAM LOB27;   {LFA 27, 2 cullsizes, 6 Jan 88, proportional moult}
{$R+,O+}
uses Dos,Printer,crt,listnums;
       const
          amoultM : real = -0.27;          {all Miller etc 87}
          bmoultM : real = 1.15;
          amoultF : real = 9.67;
          bmoultF : real = 1.0;
          apropM : real = 3.77;
          bpropM : real = -0.019;
          apropF : real = 3.77;
          bpropF : real = -0.019;
          awtM : real = 0.000566;
          bwtM : real = 3.078;
          awtF : real = 0.001525;
          bwtF : real = 2.8612;
          M_annual : real = 0.1;
          Firstyear = 1989;
          Lastyear = 1998;
          startsize = 70;
          maxage = 5;
          maxsize = 160;
        Type
          agesizeA = array[1..maxage,startsize..maxsize] of real;
          sizeVec = array[startsize..maxsize] of real;
        Var
          ch,print_init_sizes,print_final_sizes : char;
          age,cullsize,day,i,j,finish,nexts,season,size,
           closeday,new_cullsize,old_cullsize,openday,start,year : integer;
          Fday,F_simulation,Mday,Z,Pre_rate_of_exploitation,
                                              rate_of_exploitation : real;
          Males,Immature,Mature,Berried,popfreq : agesizeA;
          Pmature,propM,propF,weightM,weightF,yieldM,yield1,yield0:sizeVec;
          nextF,nextM : array[startsize..maxsize] of integer;
          tape1,tape2 : text;
          yr,mon,dy,dywk,hr,min,sec,sec100 : word;
          filename1 : string[11];
          run_name : string[80];
-------------------------------------------------------------------------
PROCEDURE SET_YIELDS_TO_ZERO;
BEGIN
For size := startsize to maxsize Do
  BEGIN
  yieldM[size] := 0;
  yield1[size] := 0;
  yield0[size] := 0;
```

```
 END;  {size}
END;
-------------------------------------------------------------------------------
PROCEDURE GET_PARAMETERS;
BEGIN
ch := 'p';
Repeat
  Clrscr;
  GoToxy(10,1);
  writeln('Enter name of simulation for LFA 27');
  writeln;
  readln(run_name);
  GoToxy(2,5);
  write('Enter Julian date of opening day of season ?');
  Gotoxy(62,5);
  readln(openday);
  Gotoxy(2,7);
  write('Enter Julian date of closing day of season? ');
  Gotoxy(62,7);
  readln(closeday);
  Gotoxy(2,9);
  write('Enter rate of exploit. during previous time period ?');
  gotoxy(62,9);
  readln(Pre_rate_of_exploitation);
  gotoxy(2,11);
  write('Enter annual rate of exploitation for the simulation ?');
  gotoxy(62,11);
  readln(rate_of_exploitation);
  gotoxy(2,13);
  write('Enter old cull size in millimetres ?');
  gotoxy(62,13);
  readln(old_cullsize);
  if old_cullsize < startsize Then
  Begin
  writeln('startsize to big, change startsize in constant declarations');
  halt;
  end;
  gotoxy(2,15);
  write('Enter future cull size in millimetres ?');
  gotoxy(62,15);
  readln(new_cullsize);
  if new_cullsize < startsize Then
  Begin
  writeln('startsize to big, change startsize in constant declarations');
  halt;
  end;
  gotoxy(2,17);
  writeln('Enter output file name as abcdefgh.abc ?');
  gotoxy(62,17);
  readln(filename1);
  Assign(tape2,filename1);
  Rewrite(tape2);
  writeln(tape2,run_name);
  gotoxy(2,19);
```

```
  write('Enter y if you want a list of the initial size frequencies ?');
  Gotoxy(62,19);
  readln(print_init_sizes);
  Gotoxy(2,21);
  write('Enter y if you want a list of the final size frequencies ?');
  Gotoxy(62,21);
  readln(print_final_sizes);
  Gotoxy(2,23);
  writeln('Enter any character to proceed; enter "B" if there is an
error');
  readln(ch);
Until Not ((ch = 'B') or (ch = 'b'));
end;
-------------------------------------------------------------------------
PROCEDURE SET_VARIABLES;
Begin
  For size := startsize to maxsize Do
  Begin
  NextM[size] := round(exp(ln(amoultM)+bmoultM * Ln(size)));
  NextF[size] := round(exp(ln(amoultF) + bmoultF * Ln(size)));
  propM[size] := apropM * exp(bpropM * size);
  If propM[size] > 1 Then propM[size] := 1;
  propF[size] := apropF * exp(bpropF * size);
  If propF[size] > 1 Then propF[size] := 1;
  Pmature[size] := 1 / (1 + exp(21.8 - size * 0.258));
  weightM[size] :=(exp(Ln(awtM) + bwtM * Ln(size)))/1000;
  weightF[size] :=(exp(Ln(awtF) + bwtF * Ln(size)))/1000;
  End;              {size}
F_simulation := Ln(1/(1-Rate_of_Exploitation));
Mday := M_annual/365;
If openday > closeday then season := closeday + 365 + 1 - openday
  Else season := 1 + closeday - openday;
writeln('length of season = ',season,' days');
Fday := F_simulation/season
End;  {set_variables}
-------------------------------------------------------------------------
PROCEDURE  RECRUIT;        {sets  up  1st  age  class  size  change  only if
nonconstant}
Begin
For size := startsize to (nextM[startsize] - 1) Do
                    Males[1,size] := 10000/(nextM[startsize] - startsize);
For size := nextM[startsize]  to maxsize Do Males[1,size] := 0;
For size := startsize to (nextF[startsize] - 1) do
  Begin
  Immature[1,size] := 10000/(nextF[startsize] - startsize);
  Mature[1,size] := Immature[1,size] * Pmature[size];
  Immature[1,size] := Immature[1,size] - Mature [1,size];
  End;
For size := nextF[startsize] to maxsize do
  Begin
  Immature[1,size] := 0;
  Mature[1,size] := 0;
  End;
End;
```

```
-------------------------------------------------------------------------
PROCEDURE OVIGEROUS;
Var
  expected_mature,new_mature : real;
Begin
expected_mature:=Pmature[size]*(Immature[age,size]+Mature[age,size]
                                                    + Berried[age,size]);

If expected_mature > Berried[age,size] + Mature[age,size] Then
  Begin
  new_mature := expected_mature - Mature[age,size] - Berried[age,size];
  Mature[age,size] := new_mature + Mature[age,size];
  Immature[age,size]:=Immature[age,size]-new_mature;
  END;
END;  {ovigerous}
-------------------------------------------------------------------------
PROCEDURE SET_POP_ARRAYS_TO_ZERO;
Begin
  For size := startsize to maxsize Do
  Begin
  For age := 1 to maxage Do
    Begin
    Males[age,size] := 0;
    Immature[age,size] := 0;
    Mature[age,size] := 0;
    Berried[age,size] := 0;
    End;              {age}
  End;               {size}
End;  {set_pop_arrays_to_zero}
-------------------------------------------------------------------------
PROCEDURE INITIALISE_MALES;
Var
  nexts : integer;
  Fday,survivors : real;
Begin
Fday := (Ln(1/(1-pre_rate_of_exploitation)))/season;
For age := 2 to Maxage Do
  Begin
  For size := startsize to maxsize do
    Begin
    Nexts := NextM[size];
    If Males[age-1,size] > 0 Then
      Begin
      survivors := Males[age-1,size] *  exp(season*(-Fday - Mday));
      survivors := survivors * exp((365-season) * -Mday);
      Males[age,size] := Males[age,size] + survivors * (1 - propM[size]);
      If Nexts <= maxsize Then
           Males[age,nexts] := Males[age,nexts] + survivors * propM[size];
      End;              { males > 0 }
    End;                {size}
  end;        {age}
END;   {init males}
-------------------------------------------------------------------------
PROCEDURE INITIALISE_FEMALES;
```

```
Var
  nexts : integer;
  Fday,survivors : real;
Begin
Fday := Ln(1/(1-pre_rate_of_exploitation))/season;
For age := 2 to maxage Do
  Begin
  For size := startsize to maxsize do
    Begin
    Nexts := NextF[size];
{bring immature through}
    If Immature[age-1,size] > 0 Then
      Begin
      survivors := Immature[age-1,size] * exp(season*(-Fday-Mday));
      survivors := survivors * exp((365-season) * -Mday);
      Immature[age,size]:=Immature[age,size]+ survivors * (1 - propF[size]);
      If Nexts <= maxsize Then
        Immature[age,nexts] := Immature[age,nexts] + survivors *propF[size];
      End;    {immature>0}
 {Bring Berried through}
    If Berried[age-1,size] > 0 Then
      Begin
      survivors := Berried[age-1,size] * exp(365 * -Mday);
      Mature[age,size] := Mature[age,size] + survivors * (1 - propF[size]);
      If Nexts <= maxsize Then
        Mature[age,nexts] := Mature[age,nexts] + survivors * propF[size];
      End;   {berried>0}
{bring mature through}
    If Mature[age-1,size] > 0 Then
      Begin
      survivors := Mature[age-1,size] * exp(season*(-Fday-Mday));
      survivors := survivors * exp((365-season) * -Mday);
      Berried[age,size] := Berried[age,size] + survivors * (1 -propF[size]);
      If Nexts <= maxsize Then
            Berried[age,nexts]:=Berried[age,nexts]+survivors * propF[size];
      End;    {mature>0}
    If immature[age,size] > 0 Then OVIGEROUS;
    END;   {size}
   END;  {for age}
end;   {initialise_numbers}
-------------------------------------------------------------------------
PROCEDURE HARVESTM (Fmort : real);
var
  Fday,numdead : real;
Begin
  For age := 1 to maxage Do
    Begin
    For size := startsize to maxsize Do
      Begin
      If Males[age,size]>0 Then
        Begin
        If size < cullsize Then Fday := 0 Else Fday := Fmort;
        numdead := Males[age,size] * (1 - exp(season*(-Fday-Mday)));
        yieldM[size]:=yieldM[size]+(Fday/(Fday+Mday))*numdead*weightM[size];
```

```
        Males[age,size] := Males[age,size] - numdead;
        Males[age,size] := Males[age,size] *exp((365-season)*-Mday);
        End;   {males>0}
      End;   {for size}
    End;     {for age}
end;   {harvestM}
-------------------------------------------------------------------------
PROCEDURE HARVESTF (Fmort : real);
var
  Fday,numdead : real;
begin
  For size := startsize to maxsize Do
  Begin
  If size < cullsize Then Fday := 0 Else Fday := Fmort;
    For age := 1 to maxage Do
    Begin
    numdead := Immature[age,size] * (1 - exp(season*(-Fday-Mday)));
    yieldI[size]:=yieldI[size]+ (Fday/(Fday+Mday))*numdead*weightF[size];
    Immature[age,size] := Immature[age,size] - numdead;
    Immature[age,size] := Immature[age,size] * exp((365-season)*-Mday);
    numdead := Mature[age,size] * (1 - exp(season*(-Fday-Mday)));
    yield0[size]:=yield0[size]+(Fday/(Fday+Mday)) * numdead *weightF[size];
    Mature[age,size] := Mature[age,size] - numdead;
    Mature[age,size] := Mature[age,size] * exp((365-season)*-Mday);
    Berried[age,size] := Berried[age,size] * exp(-Mday*365);
    End;   {for age}
  End;     {for size}
 end;   {harvestF}
-------------------------------------------------------------------------
PROCEDURE HARVEST_TO_FILE;
var
  size : integer;
Begin
If year = firstyear Then
  writeln(tape2 ,startsize:4,cullsize:4,maxsize:4,firstyear:6,lastyear:6,
                                                Rate_of_Exploitation:8:2);
writeln(tape2,year);
writeln(tape2,' CL      Males     Immature    Mature     total');
For size := startsize to maxsize Do
  writeln(tape2,size:3,yieldM[size]:10:1,yieldI[size]:10:1,yield0[-
size]:10:1);
End;
-------------------------------------------------------------------------
PROCEDURE MOULT;
Var
  nexts : integer;
Begin
For age := maxage DOWNTO 2 Do
  Begin
  For size := startsize to maxsize Do Males[age,size] := 0;
  For size := startsize to maxsize Do
    Begin
    nexts := nextM[size];
    If Males[age-1,size] > 0 Then
```

```
      Begin
      Males[age,size]:=Males[age,size]+Males[age-1,size] * (1 -propM[size]);
      If nexts <= maxsize Then
       Males[age,nexts]:=Males[age,nexts]+ Males[age-1,size] * propM[size];
      End;  { males > 0 }
    end;    { size }
  For size := startsize to maxsize Do Immature[age,size] := 0;
  For size := startsize to maxsize Do
    Begin
    If Immature[age-1,size] > 0 Then
      Begin
      nexts:= nextF[size];
      Immature[age,size] :=  Immature[age,size] + Immature[age-1,size]
                                              * (1-propF[size]);
      If nexts <= maxsize Then
       Immature[age,nexts] := Immature[age,nexts] + Immature[age-1,size]
                                                  * propF[size];
      End;      { num > 0 }
    end; {size}
  For size := startsize to maxsize Do Mature[age,size] := 0;
  For size := startsize to maxsize Do
    Begin
    If Berried[age-1,size] > 0 Then
      Begin
      Nexts:= nextF[size];
      Mature[age,size] := Mature[age,size] + Berried[age-1,size] *
                                                  (1-propF[size]);
      If nexts <= maxsize Then
       Mature[age,nexts]:=Mature[age,nexts]+Berried[age-1,size]*propF[size];
      End;    {berried>0}
    end; {size}
  For size := startsize to maxsize Do Berried[age,size] := 0;
  For size := startsize to maxsize Do
    Begin
    Nexts:= nextF[size];
    If Mature[age-1,size] > 0  Then
      Begin
      Berried[age,size] := Berried[age,size] + Mature[age-1,size] *
                                                  (1 - propF[size]);
      If nexts <= maxsize Then
      Berried[age,nexts]:=Berried[age,nexts]+Mature[age-1,size]*propF[size];
      End;      { mature > 0 }
    If Immature[age,size] > 0 Then OVIGEROUS;
    end;  { size }
  End;  {age}
End; {MOULT}
-------------------------------------------------------------------------
PROCEDURE LIST_AGESIZE (Var poptreq : agesizeA; ch : char);
Var
  i,size :integer;
  tot : real;
 Begin
  Case ch Of
  'M' : write(lst,'         Male age composition ');
```

```
  'I' : write(lst,'        Immature Female age composition ');
  'O' : write(lst,'        Mature Female age composition ');
  'B' : write(lst,'        Berried Female age composition ');
  End;
Writeln(lst,': ',year);
Writeln(lst,' CL      1         2         3         4         5       total');
  For size := startsize to maxsize Do
  Begin
    write(lst,size:4);
    For i := 1 to maxage - 1 Do
    write(lst,popfreq[i,size]:9:2);
    write(lst,popfreq[maxage,size]:9:2);
    tot := 0;
    For i := 1 to maxage Do tot := tot + popfreq[i,size];
    writeln(lst,tot:9:2);
  end;
End;       {listfreq}
-------------------------------------------------------------------------------
PROCEDURE WRITENUMS;
BEGIN
ch := 'M';
LIST_AGESIZE (Males,ch);
ch := 'I';
LIST_AGESIZE (Immature,ch);
ch := 'O';
LIST_AGESIZE (Mature,ch);
ch := 'B';
LIST_AGESIZE (Berried,ch);
END;   {Writenums}
Begin                                          {main procedure}
GET_PARAMETERS;
SET_VARIABLES;
SET_POP_ARRAYS_TO_ZERO;
RECRUIT;
INITIALISE_MALES;
INITIALISE_FEMALES;
{If (print_init_sizes = 'y') or (print_init_sizes = 'Y') Then WRITENUMS;}
For year := firstyear to lastyear Do
  BEGIN
  cullsize := old_cullsize;
  SET_YIELDS_TO_ZERO;
  If year > firstyear Then RECRUIT;
  writeln('year = ',year,' cullsize = ',cullsize);
  HARVESTM(Fday);
  HARVESTF(Fday);
  HARVEST_TO_FILE;
  MOULT;
  End;  {year}
For year := firstyear to lastyear Do
  BEGIN
  cullsize := new_cullsize;
  SET_YIELDS_TO_ZERO;
  RECRUIT;
  writeln('year = ',year,' cullsize = ',cullsize);
```

```
  HARVESTM(Fday);
  HARVESTF(Fday);
  HARVEST_TO_FILE;
  MOULT;
  End;  {year}
If (print_final_sizes = 'y') or (print_final_sizes = 'Y') Then  WRITENUMS;
close (tape2);
writeln('finished')
end.
```

Appendix II

Revenues Program.

```
PROGRAM LOB2S3; {processes data from LOB1 - 2 cull sizes}
{$R+,U+}
uses Printer,crt;
const
  discount = 0.1;
type sizevec = array[60..200] of real;
var
  cullsize,maxsize,number_of_years,size,year : integer;
  lasty,total,twenty,total_discounted,rate_of_Exploitation,sim_year,
                                    total_nondis,totyearW : real;
  yieldM,yieldI,yield0 : sizevec;
  tape1,tape2 : text;
  cannerW,chixw,quartw,halfw,marketW,selectw,jumboW,cannerR,chixR,-
quartR,halfR,
    selectR,jumboR,totalM,year_discounted,totalI,total0   :   array[1..20]   of
real;
  filename1 : string[11];
 run_name : string[80];
-------------------------------------------------------------------------
PROCEDURE GET_FILE;
BEGIN
writeln('enter input file name');
readln(filename1);
writeln(Lst,'file being processed = ',filename1);writeln(lst);
Assign(Lst,filename1);
Append(Lst);Reset(tape1);
Readln(Lst,run_name);
writeln(run_name);
END;
-------------------------------------------------------------------------
PROCEDURE SETVARIABLES;
var
  year : integer;
Begin
For year := 1 to 20 Do
  Begin
  totalM[year] := 0;
  totalI[year] := 0;
  total0[year] := 0;
  chixw[year] := 0;
  quartw[year] := 0;
  halfw[year] := 0;
  selectw[year] := 0;
  jumbow[year] := 0;
  cannerW[year] := 0;
  marketW[year] := 0;
  End;
```

```
End;
-------------------------------------------------------------------------
PROCEDURE GET_DATA;
var
  annual,cl,minsize,size,yearstop,yearstart : integer;
  total : real;
  line : string[40];
Begin
Readln(tape1,minsize,cullsize,maxsize,yearstart,yearstop,rate_of-
_Exploitation);
If maxsize > 200 Then
  Begin
  write('maximum size > 200, fix !!!!!!!'); halt;
  End;
number_of_years := 1 + yearstop - yearstart;
If number_of_years > 10 Then
  Begin
  write('more than 10 years, fix!!!!!!!!!'); halt;
  End;
  For year := 1 to number_of_years Do
    BEGIN
    readln(tape1,sim_year);
    Readln(tape1,line);
    For size := minsize to maxsize Do
      Begin
      Readln(tape1,cl,yieldM[size],yieldI[size],yieldO[size]);
      totalM[year] := totalM[year] + yieldM[size];
      totalI[year] := totalI[year] + yieldI[size];
      totalO[year] := totalO[year] + yieldO[size];
      End;
    If minsize < 81 Then
      For size := minsize to 80 Do
       cannerw[year]    :=    cannerw[year]+yieldM[size]+    yieldI[size]    +
yieldO[size];
    For size := 81 to 89 Do
       chixw[year]   :=   chixw[year]   +   yieldM[size]   +   yieldI[size]   +
yieldO[size];
    For size := 90 to 95 Do
     quartw[year]   :=   quartw[year]   +   yieldM[size]   +   yieldI[size]   +
yieldO[size];
    For size := 96 to 104 Do
        halfw[year]:=halfw[year]    +    yieldM[size]    +    yieldI[size]   +
yieldO[size];
    For size := 105 to 119 Do
   selectw[year]   :=   selectw[year]   +   yieldM[size]   +   yieldI[size]   +
yieldO[size];
    For size := 120 to maxsize Do
     jumbow[year]   :=   jumbow[year]   +   yieldM[size]   +   yieldI[size]   +
yieldO[size];
  End;   {year}
End;
-------------------------------------------------------------------------
PROCEDURE REVENUES;
const
```

```
  cannerP : real = 2.81;
  chixP : real = 3.75;
  quartP : real = 4.0;
  halfP : real = 6.0;
  selectP : real = 7.0;
  jumboP : real = 6.5;
Begin
For year := 1 to number_of_years Do
  Begin
  cannerR[year] := cannerP * cannerW[year];
  chixR[year] := chixP * chixw[year];
  quartR[year] := quartP * quartw[year];
  halfR[year] := halfP * halfW[year];
  selectR[year] := selectP * selectW[year];
  jumboR[year] := jumboP * jumbow[year];
  End;
  total_discounted := 0;
  total_nondis := 0;
writeln(Lst,'cullsize= ',cullsize,'                    REVENUES');
writeln(Lst,'Year Canners   Chix  Quarters  Halves  Selects    Jumbos     Total
Discounted');
For year := 1 to number_of_years Do
  Begin
  total:= cannerR[year] + chixR[year] + quartR[year] +
                              halfR[year] + selectR[year] + jumboR[year];
  total_nondis := total_nondis + total;
  year_discounted[year] := total / exp((year - 1) * ln (1 + discount));
  total_discounted := total_discounted + year_discounted[year];
  writeln(Lst,year:3,cannerR[year]:8:0,chixR[year]:8:0,quartR[year]:8:0,
halfR[year]:8:0,selectR[year]:8:0,jumboR[year]:8:0,total:9:0,yea-
r_discounted[year]:9:0);
  End;
writeln(Lst,'                                                          _______
_____');
writeln(Lst,'
',total_nondis:16:0,total_discounted:9:0);
lasty := year_discounted[year];
twenty := 0;
for year := 1 to 10 Do twenty := twenty + lasty / exp(year * ln(1 + discount));

writeln(Lst,'Total discounted over a twenty year  period .....
_____');
twenty := twenty + total_discounted;
writeln(Lst,'
',twenty:16:0);

End;
-------------------------------------------------------------------------
PROCEDURE WEIGHT;
var
  total,total_weight : real;
Begin
total_weight := 0;
writeln(Lst,'                                WEIGHT');
```

```
writeln(Lst,'Year  Canners  Chix Quarters  Halves Selects  Jumbos   Total');
For year := 1 to number_of_years Do
  Begin
  total:=cannerW[year] + chixW[year] + quartW[year]
                                +    halfW[year]     +     selectW[year]    +
jumboW[year];
  total_weight := total_weight + total;
  writeln(Lst,year:3,cannerW[year]:8:0,chixW[year]:8:0,quartW[year]:8:0,

halfW[year]:8:0,selectW[year]:8:0,jumboW[year]:8:0,total:9:0);
  End;
writeln(Lst,'                                                          _______');
writeln(Lst,'                                                ',total_weight:16:0);
writeln(Lst,' ');writeln(tape2,' ');
End;
{-------------------------------------------------------------------------------}
Begin
GET_FILE;
SETVARIABLES;
GET_DATA;
REVENUES;
WEIGHT;
SETVARIABLES;
GET_DATA;
REVENUES;
WEIGHT;
Close(tape1);
writeln('finished')
end.
```

Appendix IIIa

Code for "Rolling-wave" moult model.

```
PROGRAM LOBI;
{$R+,U+}
uses Printer,crt,listnums,setvars;
-------------------------------------------------------------------
PROCEDURE SET_YIELDS_TO_ZERO;
BEGIN
For size := startsize to maxsize Do
  BEGIN
  yieldM[size] := 0;
  yieldI[size] := 0;
  yieldO[size] := 0;
  END;  {size}
END;
-------------------------------------------------------------------
PROCEDURE SET_VARIABLES;
Begin
  For size := startsize to maxsize Do
  Begin
  NextM[size] := round(exp(ln(amoultM)+bmoultM * Ln(size)));
  NextF[size] := round(exp(ln(amoultF) + bmoultF * Ln(size)));
  propM[size] := apropM * exp(bpropM * size);
  If propM[size] > 1 Then propM[size] := 1;
  propF[size] := apropF * exp(bpropF * size);
  If propF[size] > 1 Then propF[size] := 1;
  Pmature[size] := 1 / (1 + exp(23.23 - size * 0.214));
  weightM[size] :=(exp(Ln(awtM) + bwtM * Ln(size)))/1000;
  weightF[size] :=(exp(Ln(awtF) + bwtF * Ln(size)))/1000;
  End;                {size}
writeln('Enter output file name');
readln(filename1);
Assign(tape2,filename1);
Rewrite(tape2);
write('enter rate of exploitation ');
readln(rate_of_exploitation);
F_simulation := Ln(1/(1-Rate_of_Exploitation));
Mday := M_annual/365;
If openday > closeday then season := closeday + 365 + 1 - openday
                          Else season := 1 + closeday - openday;
writeln('length of season = ',season,' days');
Fday := F_simulation/season
End;  {set_variables}
-------------------------------------------------------------------
PROCEDURE  RECRUIT;        {sets  up  1st  age class size change only if
nonconstant}
Begin
```

```
For size := startsize to (nextM[startsize] - 1) Do
  Begin
  Males[1,size] := 10000/(nextM[startsize] - startsize);
  transM[1,size] := Males[1,size] * propM[size];
  End;
For size := nextM[startsize]  to maxsize Do
  Begin
  Males[1,size] := 0;
  transM[1,size] := 0;
  End;
For size := startsize to (nextF[startsize] - 1) do
  Begin
  Immature[1,size] := 10000/(nextF[startsize] - startsize);
  Mature[1,size] := Immature[1,size] * Pmature[size];
  Immature[1,size] := Immature[1,size] - Mature [1,size];
  transI[1,size] := propF[size] * Immature[1,size];
  transO[1,size] := propF[size] * Mature[1,size];
  End;
For size := nextF[startsize] to maxsize do
  Begin
  Immature[1,size] := 0;
  Mature[1,size] := 0;
  transI[1,size] := 0;
  transO[1,size] := 0;
  End;
End;
-------------------------------------------------------------------
PROCEDURE OVIGEROUS;
Var
  expected_mature,new_mature : real;
Begin
expected_mature := Pmature[size] * (Immature[age,size] +
                                   Mature[age,size]  + Berried[age,size]);
If expected_mature > Berried[age,size] + Mature[age,size] Then
  Begin
  new_mature := expected_mature - Mature[age,size] - Berried[age,size];
  Mature[age,size] := new_mature + Mature[age,size];
  transO[age,size] := transO[age,size] + (new_mature/Immature[age,size])
                                                  * transI[age,size];
 transI[age,size] := transI[age,size] - (new_mature/Immature[age,size])
                                                  * transI[age,size];
  Immature[age,size]:=Immature[age,size]-new_mature;
  END;
END;  {ovigerous}
-------------------------------------------------------------------
PROCEDURE SET_POP_ARRAYS_TO_ZERO;
Begin
  For size := startsize to maxsize Do
  Begin
  For age := 1 to maxage Do
    Begin
    Males[age,size] := 0;
```

```
    Immature[age,size] := 0;
    Mature[age,size] := 0;
    Berried[age,size] := 0;
    transM[age,size] := 0;
    transI[age,size] := 0;
    transO[age,size] := 0;
    transB[age,size] := 0
    End;              {age}
  End;              {size}
End;  {set_pop_arrays_to_zero}
-------------------------------------------------------------------
PROCEDURE INITIALISE_MALES;
Var
  nexts : integer;
  Fan,survivors,moulters : real;
Begin
{Set up age-size frequency distribution}
Fan := Ln(1/(1-initial_Fmort));
For age := 2 to Maxage Do
  Begin
  For size := startsize to maxsize do
    Begin
    Nexts := NextM[size];
    If Males[age-1,size] > 0 Then
      Begin
      survivors := Males[age-1,size] *  exp(season*(-Fday - Mday));
      moulters := transM[age-1,size] * exp(season * (-Fday - Mday));
      survivors := survivors * exp((365-season) * -Mday);
      moulters := moulters * exp((365-season) * -Mday);
      Males[age,size] := Males[age,size] + survivors - moulters;
      If moulters < 0.5 * survivors
        Then transM[age,size] := transM[age,size] + moulters Else
        If moulters < survivors Then transM[age,size]:=transM[age,size]
                                                + survivors - moulters;
     If 1E-10 < moulters - survivors Then
                    writeln('IN M m>s ',size,' ',moulters - survivors);
      If Nexts <= maxsize Then
        Begin
        Males[age,nexts] := Males[age,nexts] + moulters;
        transM[age,nexts] := transM[age,nexts] + propM[nexts] *
                                                  moulters;
        END;
      End;
    End;              {size}
  end;        {age}
END;   {init males}
-------------------------------------------------------------------
PROCEDURE INITIALISE_FEMALES;
Var
  nexts : integer;
  Fan,survivors,moulters : real;
Begin
```

```
{Set up age-size frequency distribution}
Fan := Ln(1/(1-initial_Fmort));
  For age := 2 to maxage Do
    Begin
    For size := startsize to maxsize do
      Begin
      Nexts := NextF[size];
{bring immature through}
    If Immature[age-1,size] > 0 Then
      Begin
      survivors := Immature[age-1,size] * exp(season*(-Fday-Mday));
      moulters := transI[age-1,size] * exp(season*(-Fday-Mday));
      survivors := survivors * exp((365-season) * -Mday);
      moulters := moulters * exp((365-season) * -Mday);
      Immature[age,size] := Immature[age,size] + survivors - moulters;
      If moulters < 0.5 * survivors
        Then transI[age,size] := transI[age,size] + moulters Else
        If moulters < survivors Then transI[age,size]:=transI[age,size]
                                                + survivors - moulters;
     If 1E-10< moulters-survivors Then writeln('IN IM m>s ',size,'
                                                ',moulters - survivors);
      If Nexts <= maxsize Then
        Begin
        Immature[age,nexts] := Immature[age,nexts] + moulters;
        transI[age,nexts]:= transI[age,nexts] + propF[nexts] * moulters;
        END;
      End;    {immature>0}
 {Bring Berried through}
    If Berried[age-1,size] > 0 Then
      Begin
      survivors := Berried[age-1,size] * exp(365 * -Mday);
      moulters := transB[age-1,size] * exp(365 * -Mday);
      Mature[age,size] := Mature[age,size] + survivors - moulters;
      If moulters < 0.5 * survivors
        Then transO[age,size] := transO[age,size] + moulters Else
        If moulters < survivors Then transO[age,size]:=transO[age,size]
                                                + survivors - moulters;
     If 1E-10<  moulters-survivors Then writeln('IN B m>s ',size,'
                                               ',moulters,' ',survivors);
     If Nexts <= maxsize Then
       Begin
       Mature[age,nexts] := Mature[age,nexts] + moulters;
       transO[age,nexts]:=transO[age,nexts] + propF[nexts] * moulters;
       End;
     End;   {berried>0}
{bring mature through}
    If Mature[age-1,size] > 0 Then
      Begin
      survivors := Mature[age-1,size] * exp(season*(-Fday-Mday));
      moulters := transO[age-1,size] * exp(season*(-Fday-Mday));
      survivors := survivors * exp((365-season) * -Mday);
      moulters := moulters * exp((365-season) * -Mday);
```

```
      Berried[age,size] := Berried[age,size] + survivors - moulters;
      If moulters < 0.5 * survivors
        Then transB[age,size] := transB[age,size] + moulters Else
        If moulters < survivors Then transB[age,size]:=transB[age,size]
                                                + survivors - moulters;
     If 1E-10< moulters-survivors Then writeln('IN O m>s ',size,'
                                              ',moulters,' ',survivors);
      If Nexts <= maxsize Then
        Begin
        Berried[age,nexts] := Berried[age,nexts] + moulters;
        transB[age,nexts]:=transB[age,nexts] + propF[nexts] * moulters;
        End;
      End;    {mature>0}
    If immature[age,size] > 0 Then OVIGEROUS;
    End;   {size}
   End;  {for age}
End;   {initialise_numbers}
-------------------------------------------------------------------
PROCEDURE HARVESTM (Fmort : real);
var
  Fday,numdead : real;
Begin
  For age := 1 to maxage Do
    Begin
    For size := startsize to maxsize Do
      Begin
      If Males[age,size]>0 Then
        Begin
        If size < cullsize Then Fday := 0 Else Fday := Fmort;
        numdead := Males[age,size] * (1 - exp(season*(-Fday-Mday)));
        yieldM[size]:=yieldM[size]+(Fday/(Fday +
                                          Mday))*numdead*weightM[size];
        Males[age,size] := Males[age,size] - numdead;
        transM[age,size] := transM[age,size]*exp(season*(-Fday - Mday));
        Males[age,size] := Males[age,size] *exp((365-season)*-Mday);
        transM[age,size] := transM[age,size]*exp((365-season)*-Mday);
        End;   {males>0}
      End;   {for size}
    End;     {for age}
end;   {harvestM}
-------------------------------------------------------------------
PROCEDURE HARVESTF (Fmort : real);
var
  Fday,numdead : real;
begin
  For size := startsize to maxsize Do
  Begin
  If size < cullsize Then Fday := 0 Else Fday := Fmort;
    For age := 1 to maxage Do
    Begin
    numdead := Immature[age,size] - Immature[age,size]
                                    * exp(season * ( -Fday -Mday));
```

```
    yield1[size] := yield1[size] + (Fday / (Fday + Mday)) *
                                                numdead * weightF[size];
    Immature[age,size] := Immature[age,size] - numdead;
    transI[age,size] := transI[age,size] * exp(season * (-Fday-Mday));
If 1E-10 < transI[age,size] - Immature[age,size] Then
    writeln('HARV ',age,' ',size,' ',Immature[age,size],' ',
                                                    transI[age,size]);
    Immature[age,size] := Immature[age,size] * exp((365-season)*-Mday);
    transI[age,size] := transI[age,size] * exp((365-season)*-Mday);
    numdead :=  Mature[age,size]-Mature[age,size] *
                                          exp(season *( -Fday - Mday));
     yield0[size] := yield0[size] + (Fday/(Fday + Mday)) * numdead
                                                  * weightF[size];
   Mature[age,size] := Mature[age,size] - numdead;
   Trans0[age,size] := trans0[age,size] * exp(season * (-Fday - Mday));
    Mature[age,size] := Mature[age,size] * exp((365 - season) * -Mday);
    trans0[age,size] := trans0[age,size] * exp((365 - season) * -Mday);
    Berried[age,size] := Berried[age,size] * exp(-Mday * 365);
    transB[age,size] := transB[age,size] * exp(-Mday * 365);
    End;   {for age}
  End;     {for size}
 end;   {harvestF}
-------------------------------------------------------------------
PROCEDURE HARVEST_TO_FILE;
var
  size : integer;
Begin
If year = firstyear Then
  writeln(tape2,startsize:4,cullsize:4,maxsize:4,firstyear:6,lastyear:6,
                                                Rate_of_Exploitation:8:2);
writeln(tape2,year);
writeln(tape2,' CL     Males    Immature   Mature    total');
For size := startsize to maxsize Do
  writeln(tape2,size:3,yieldM[size]:10:1,yield1[size]:10:1,
                                                    yield0[size]:10:1);
End;
PROCEDURE MOULTM;
Var
  age,nexts : integer;
Begin
 For age := maxage DOWNTO 2 Do
  Begin
  For size := startsize to maxsize Do Males[age,size] := 0;
  For size := startsize to maxsize Do transM[age,size] :=0;
  For size := startsize to maxsize Do
   Begin
   nexts := nextM[size];
   If Males[age-1,size] > 0 Then
     Begin
     If transM[age-1,size] < 0.5 * Males[age-1,size] Then
       Begin
       If nexts <= maxsize Then
```

```
        Males[age,nexts] :=  Males[age,nexts] + transM[age-1,size];
        Males[age,size] := Males[age,size] + Males[age-1,size] -
                                                  transM[age-1,size];
        If nexts <= maxsize Then
        transM[age,nexts] := transM[age,nexts] +
                                  transM[age-1,size] * propM[nexts];
        transM[age,size] := transM[age,size] + transM[age-1,size];
        End
      Else
        BEGIN
        If transM[age-1,size] < Males[age-1,size] then
          Begin
          If nexts <= maxsize Then
          Males[age,nexts] := Males[age,nexts] + transM[age-1,size];
          Males[age,size] :=  Males[age,size] + Males[age-1,size] -
                                                  transM[age-1,size];
          If nexts <= maxsize Then
          transM[age,nexts] := transM[age,nexts]
                               + transM[age-1,size] * propM[nexts];
          transM[age,size] :=  transM[age,size] + Males[age-1,size] -
                                                  transM[age-1,size];
          End;
        END;  {else}
      If 1E-10 < transM[age-1,size] - Males[age-1,size] Then
         writeln('MM m>s ',size,'',transM[age-1,size]-Males[age-1,size]);
      If (transM[age-1,size] >= Males[age-1,size]) AND
                                              (nexts <= maxsize) Then
        Begin
        Males[age,nexts] := Males[age,nexts] + Males[age-1,size];
        transM[age,nexts] := transM[age,nexts] + Males[age-1,size] *
                                                    propM[nexts];
        End;
      End;  { num > 0}
      End;  {size}
    End;  {age}
end;  {moultM}
-------------------------------------------------------------------------
  PROCEDURE MOULT_IMMATURE;
var
  age,nexts : integer;
Begin
  For age := maxage Downto 2 Do
  Begin
    For size := startsize to maxsize Do immature[age,size] := 0;
    For size := startsize to maxsize Do trans1[age,size] := 0;
    For size := startsize to maxsize Do
      Begin
      If immature[age-1,size] > 0 Then
        Begin
        nexts:= next[size];
        If trans1[age-1,size] < 0.5 * immature[age-1,size] Then
          Begin
```

```
          If nexts <= maxsize Then
           Immature[age,nexts]:=Immature[age,nexts]+transl[age-1,size];
           Immature[age,size]:=Immature[age,size]+Immature[age-1,size]
                                                  - transl[age-1,size];
           transl[age,size] := transl[age,size] + transl[age-1,size];
          If nexts <= maxsize Then
 transl[age,nexts]:=transl[age,nexts]+propF[nexts]*transl[age-1,size];
          End
        Else if transl[age-1,size] < Immature[age-1,size] then
          Begin
          If nexts <= maxsize Then
          Immature[age,nexts]:=Immature[age,nexts]+ transl[age-1,size];
          Immature[age,size]:=Immature[age,size]+  Immature[age-1,size]
                                                  - transl[age-1,size];

          If nexts <= maxsize Then
          transl[age,nexts] := transl[age,nexts] +
                                       propF[nexts] * transl[age-1,size];
          transl[age,size] := transl[age,size] + Immature[age-1,size]-
                                                    transl[age-1,size];
          End;
      If 1E-6 < transl[age-1,size] - Immature[age-1,size] Then
       writeln('MI m>s  ',age,' ',size,' ',transl[age-1,size]:7:2,'
                                          ',Immature[age-1,size]:7:2);
      If (transl[age-1,size]>=Immature[age-1,size]) AND
                                                 (nexts <= maxsize) Then
        Begin
        Immature[age,nexts]:=Immature[age,nexts]+Immature[age-1,size];
        transl[age,nexts] := transl[age,nexts] +
                                     propF[nexts] * Immature[age-1,size];
        End;
      End;
    end; {size}
  End;  {age}
 End;  {moult_immature}
PROCEDURE MOULT_BERRIED;
var
  age,nexts : integer;
Begin
For age := maxage Downto 2 Do
  Begin
  For size := startsize to maxsize Do Mature[age,size] := 0;
  For size := startsize to maxsize Do transO[age,size] :=0;
  For size := startsize to maxsize Do
     Begin
    If Berried[age-1,size] > 0 Then
      Begin
      Nexts:= nextF[size];
      If transB[age-1,size] < 0.5 * Berried[age-1,size] Then
        Begin
        Mature[age,size]:=Mature[age,size]  +Berried[age-1,size]-transB
                                                    [age-1,size];
```

```
        If nexts <= maxsize Then
          Mature[age,nexts] := Mature[age,nexts] + transB[age-1,size];
        Mature[age,size]:=Mature[age,size]+Berried[age-1,size]-  transB
                                                          [age-1,size];
       transO[age,size] := transO[age,size] + transB[age-1,size];
        If nexts <= maxsize Then
  transO[age,nexts]:=transO[age,nexts]+propF[nexts]*transB[age-1,size];
        End
       Else if transB[age-1,size] < Berried[age-1,size] Then
        Begin
        If nexts <= maxsize Then
          Mature[age,nexts] := Mature[age,nexts] + transB[age-1,size];
       Mature[age,size] := Mature[age,size] + Berried[age-1,size]
                                              - transB[age-1,size];
        If nexts <= maxsize Then
   transO[age,nexts]:=transO[age,nexts]+propF[nexts]*transB[age-1,size];
        transO[age,size] :=  transO[age,size] + Berried[age-1,size]
                                                - transB[age-1,size];
       End;
      If 1E-10 < transB[age-1,size] - Berried[age-1,size] Then
       writeln('MB m>s',size,'',transB[age-1,size]-Berried[age-1,size]);
      If (transB[age-1,size] >= Berried[age-1,size]) AND
                                              (nexts <= maxsize) Then
        Begin
        Mature[age,nexts] := Mature[age,nexts] + Berried[age-1,size];
        transO[age,nexts] := transO[age,nexts] + propF[nexts] *
                                                 Berried[age-1,size];
        End;
      End;    {berried>0}
    end; {size}
  End;  {age}
End; {MOULT_BERRIED}
-------------------------------------------------------------------------
PROCEDURE MOULT_MATURE;
var
  nexts : integer;
Begin
  For age := maxage Downto 2 Do
    Begin
    For size := startsize to maxsize Do Berried[age,size] := 0;
    For size := startsize to maxsize Do transB[age,size] :=0;
    For size := startsize to maxsize Do
      Begin
      Nexts:= next[size];
      If worktrans[age-1,size] < 0.5 * worknum[age-1,size] Then
        Begin
        If nexts <= maxsize Then
          Berried[age,nexts] := Berried[age,nexts]
                                          +worktrans[age-1,size];
         Berried[age,size] := Berried[age,size] + worknum[age-1,size]
                                         - worktrans[age-1,size];
        transB[age,size] := transB[age,size] + worktrans[age-1,size];
```

```
        If nexts <= maxsize Then
        transBlage,nexts] := transBlage,nexts] + propF[nexts]
                                              *  workTranslage-1,size];
       End
   Else  If workTranslage-1,size] < workNumlage-1,size] then
      Begin
      If nexts <= maxsize Then
      Berrledlage,nexts] := Berrledlage,nexts] + workTranslage-1,size];
      Berrledlage,size] := Berrledlage,size] + workNumlage-1,size]
                                              - workTranslage-1,size];
       If nexts <= maxsize Then
      transBlage,nexts] := transBlage,nexts] + propF[nexts] +
                                               workTranslage-1,size];
      transBlage,size] := transBlage,size] + workNumlage-1,size]
                                             -  workTranslage-1,size];
      End;
   If IE-10< workTranslage-1,size] - workTranslage-1,size] Then
    Writeln('MM     m>s      ',size,'      ',workTranslage-1,size]     >
workTranslage-1,size]);
    If (workTranslage-1,size] >= workNumlage-1,size])
                                          AND (nexts <= maxsize) Then
      Begin
      Berrledlage,nexts] := Berrledlage,nexts] + workNumlage-1,size];
      transBlage,nexts] := transBlage,nexts] + propF[nexts]
                                              * workNumlage-1,size];
      End;
    If Immature[age,size] > 0 Then OVIGEROUS;
    end; {size}
  End;  {age}
End; {MOULT_MATURE}
-------------------------------------------------------------------------
PROCEDURE WRITENUMS;    {saved to file initnum.dat}
BEGIN
ch := 'M';
LIST_AGESIZE (Males,ch);
ch := 'I';
LIST_AGESIZE (Immature,ch);
ch := 'O';
LIST_AGESIZE (Mature,ch);
ch := 'B';
LIST_AGESIZE (Berried,ch);
END;   {Writenums}
PROCEDURE DEBUG;
var
  size : integer;
  totm,toti,toto,totb,totf : real;
BEGIN
totm :=0;
toti:=0;
toto:=0;
totb:=0;
  For age := 1 to Maxage Do
```

```
    Begin
    For size := cullsize to maxsize Do
      Begin
{      totm := totm + males[age,size];
      toti := toti + immature[age,size];}
      toto := toto + mature[age,size];
      totb := totb + berried[age,size];
      end;
    end; {age}
{    totf := toti + toto + totb;}
writeln(lst,year:6,toto:11:3,totb:11:3);
END;   {debug}
Begin                                          {main procedure}
SET_VARIABLES;
For cullsize := startsize To stopsize Do
  BEGIN
  For year := firstyear to lastyear Do
    BEGIN
    SET_YIELDS_TO_ZERO;
    If year = firstyear Then SET_POP_ARRAYS_TO_ZERO;
    RECRUIT;
  writeln('year = ',year,' cullsize = ',cullsize);
    If year = firstyear Then  INITIALISE_MALES;
    If year = firstyear Then  INITIALISE_FEMALES;
{WRITENUMS;}
  HARVESTM(Fday);
  HARVESTF(Fday);
{  HARVEST_TO_FILE;}
  MOULTM;
  MOULT_IMMATURE;
  For age := 1 to Maxage Do
    Begin
    For size := startsize to maxsize Do
      Begin
      worknum[age,size] := Mature[age,size];
      worktrans[age,size] := trans0[age,size];
      End;  {size}
      End;   {age}
    MOULT_BERRIED;
    MOULT_MATURE;
    End;  {year}
  End;  {cullsize}
close (tape2);
writeln('finished')
end.
```

APPENDIX III D

This unit declares all the variables used by LOBFLSH.PAS.

```
UNIT SETVARS;
INTERFACE
        const
          amoultM : real = 1.717;
          bmoultM : real = 0.912;
          amoultF : real = 2.107;
          bmoultF : real = 0.864;
          apropM : real = 1.76;
          bpropM : real = -0.0084;
          apropF : real = 1.95;
          bpropF : real = -0.0097;
          awtM : real = 0.000566;
          bwtM : real = 3.078;
          awtF : real = 0.001525;
          bwtF : real = 2.8612;
          M_annual : real = 0.1;
          initial_Fmort : real = 0.8;
          openday :integer = 335;
          Firstyear = 1990;
          lastyear = 1995;
          closeday : integer = 152;
          startsize = 81;
          stopsize : integer = 101;
          maxage  = 8; {# year classes in fishery after recruitment}
          maxsize = 170;
        Type
          agesizeA =  array[1..maxage,startsize..maxsize] of real;
          sizeVec = array[startsize..maxsize] of real;
        Var
          ch : char;
          age,day,i,j,finish,nexts,season,size,cullsize,start,
                                                    year : integer;
          Fday,F_simulation,Mday,Z,rate_of_exploitation,
                                                  gtotm,gtotf : real;
          Males,Immature,Mature,Berried,transM,transI,transO,transB,
                                  poptreq,worknum,worktrans : agesizeA;
          Pmature,propM,propF,weightM,weightF,yieldM,yieldI,
                                                    yieldO : sizeVec;
          nextF,nextM : array[startsize..maxsize] of integer;
          tape1,tape2 : text;
          yr,mon,dy,dywk,hr,min,sec,sec100 : word;
          filename1 : string[11];
IMPLEMENTATION
End.
```

APPENDIX III C

This unit is used to list length frequency data if the calling code in LOBFLSH.PAS is invoked.

```
unit Listnums;
INTERFACE
uses setvars,crt,DOS;
PROCEDURE LIST_AGESIZE( poptreq : agesizea; var ch : char);
PROCEDURE LIST_SIZEVEC( freqvec : sizevec );
IMPLEMENTATION
PROCEDURE LIST_AGESIZE;
Var
  i,size :integer;
  tot : real;
Begin
{writeln('carapace                    age');
Writeln('length    1        2        3        4        5     total');
For size := startsize to maxsize Do
  Begin
  write(size:4);
  For i := 1 to maxage - 1 Do
  Write(poptreq[i,size]:9:3);
  Writeln(poptreq[maxage,size]:9:3);
  End;
Writeln('enter key to continue');
Repeat until Keypressed;
Read(ch);}
Assign(tape1,'initnum.dat');
GetDate(yr,mon,dy,dywk);
GetTime(hr,min,sec,sec100);
If (year = 1988) and (ch = 'M') Then Rewrite(tape1) Else Append (tape1);
If (year = 1988) and (ch = 'I') Then
                        Writeln(tape1,yr,' ',mon,' ',dy,' ',hr,' ',min);
Case ch Of
  'M' : write(tape1,'         Male age composition ');
  'I' : write(tape1,'        Immature Female age composition ');
  'O' : write(tape1,'        Mature Female age composition ');
  'B' : write(tape1,'        Berried Female age composition ');
  End;
Writeln(tape1,': ',year);
Writeln(tape1,' CL         1          2        3        4        5
total');
For size := startsize to maxsize Do
  Begin
  Write(tape1,size:4);
  For i := 1 to maxage - 1 Do Write(tape1,poptreq[i,size]:9:2);
  Write(tape1,poptreq[maxage,size]:9:2);
  tot := 0;
  For i := 1 to maxage Do tot := tot + poptreq[i,size];
```

```
  Writeln(tape1,tot:9:2);
  End;
Close (tape1);
End;        {listfreq}
PROCEDURE LIST_SIZEVEC;
Var
  size : integer;
Begin
  For size := startsize to maxsize-1 Do
  Writeln(size,' ',freqvec[size]:9:2);
End;        {list_sizevec}
End.
```

APPENDIX IV

This program calculates revenues and weights from the output from LOB1.PAS listed in Appendix III.

```
PROGRAM LOB2; {processes data from LOB1}
{$R+,O+}
uses Printer,crt;
const
  discount = 0.1;
  startsize = 81;
  stopsize = 101;
  maxsize = 170;
  firstyear = 1990;
  lastyear = 1995;
type sizevec = array[startsize..maxsize] of real;
var
  cullsize,maxsizeck,size,year : integer;
  total_discounted,rate_of_Exploitation,total,totyearW : real;
  summary_rev,yieldM,yieldI,yieldO : sizevec;
  tape1 : text;
  cannerW,chixw,quartw,haltw,marketW,selectw,jumboW,chixR,quartR,
   halfR,selectR,jumboR,totalM,year_discounted,totalI,totalO
                                    : array[firstyear..lastyear] of real;
-----------------------------------------------------------------
PROCEDURE GET_FILE;
Begin
Assign(tape1,'report.dat');
Append(tape1);Reset(tape1)
End;
-----------------------------------------------------------------
PROCEDURE SETVARIABLES;
var
  year : integer;
Begin
For year := firstyear to lastyear Do
  Begin
  totalM[year] := 0;
  totalI[year] := 0;
  totalO[year] := 0;
  chixw[year] := 0;
  quartw[year] := 0;
  haltw[year] := 0;
  selectw[year] := 0;
  jumbow[year] := 0;
  cannerW[year] := 0;
  marketW[year] := 0;
  End;
End;
-----------------------------------------------------------------
PROCEDURE GET_DATA (startsize : integer);
```

```
var
  annual,cl,minsize,size,yearstop,yearstart : integer;
  total : real;
  line : string[40];
Begin Readln(tapel,minsize,cullsize,maxsizeck,yearstart,yearstop,
                                                rate_of_Exploitation);
If cullsize <> startsize Then
  Begin
  Write('startsize does not match cullsize, fix !!!!!!!'); halt;
  End;
If maxsizeck <> maxsize Then
  Begin
  Write('maximum sizes don''t match, fix !!!!!!!'); halt;
  End;
If yearstop <> lastyear Then
  Begin
  Write('finish year <> yearf, fix!!!!!!!!!'); halt;
  End;
If yearstart <> firstyear Then
  Begin
  Write('start year <> firstyear, fix!!!!!!!!!'); halt;
  End;
  For annual := firstyear to lastyear Do
    Begin
    Readln(tapel,year);
    Readln(tapel,line);
    For size := minsize to maxsize Do
      Begin
      Readln(tapel,cl,yieldM[size],yieldI[size],yieldO[size]);
      totalM[year] := totalM[year] + yieldM[size];
      totalI[year] := totalI[year] + yieldI[size];
      totalO[year] := totalO[year] + yieldO[size];
      End;
    For size := startsize to 89 Do
       chixw[year] :=  chixw[year] + yieldM[size] + yieldI[size] +
                                                   yieldO[size];
    For size := 90 to 95 Do
     quartw[year] := quartw[year] + yieldM[size] + yieldI[size] +
                                                   yieldO[size];
    For size := 96 to 104 Do
         halfw[year]:=halfw[year] +  yieldM[size] + yieldI[size] +
                                                    yieldO[size];
    For size := 105 to 123 Do
         selectw[year] := selectw[year] + yieldM[size] +  yieldI[size] +
                                                          yieldO[size];
    For size := 124 to maxsize Do
     jumbow[year] :=  jumbow[year] + yieldM[size] + yieldI[size] +
                                                   yieldO[size];
    For size := startsize to 89 Do
       cannerw[year] := cannerw[year] + yieldM[size] + yieldI[size] +
                                                     yieldO[size];
  For size := 90 to 95 Do
```